Dennis Engler

Konzeption und Entwicklung einer Postproduction-Pipeline

Diplomarbeit

HOCHSCHULE MITTWEIDA

UNIVERSITY OF APPLIED SCIENCE

Fachbereich Elektrotechnik und Informationstechnik

Mittweida, 2010

Dennis Engler

Konzeption und Entwicklung einer Postproduction-Pipeline

eingereicht als

Diplomarbeit

an der

HOCHSCHULE MITTWEIDA

UNIVERSITY OF APPLIED SCIENCE

Fachbereich Elektrotechnik und Informationstechnik

Erstprüfer: Prof. Dr.-Ing. Robert J. Wierzbicki
Hochschule Mittweida

Zweitprüfer: Prof. Matthias Haase
CEO morro images GmbH & Co. KG

Mittweida, 2010

Engler, Dennis:
Konzeption und Entwicklung einer Postproduction-Pipeline - 2010 - 80 S.
Mittweida, Hochschule Mittweida, Fakultät Informationstechnik & Elektrotechnik,
Diplomarbeit, 2010

Referat:

Ziel der Diplomarbeit ist es, die Herangehensweise zur Entwicklung einer zukunfts-orientierten und an ein mittelständisches Unternehmen angepasste Postproduction-Pipeline aufzuzeigen.
Diese Arbeit befasst sich zunächst mit den Anforderungen, die heutzutage an eine solche branchenspezifische Managementsoftware gestellt werden. Bereits auf dem Markt vertretene Systeme werden im Folgenden untersucht. In einem weiteren Kapitel folgen die Konzeption einer für die Ziele benötigten, datenbankbasierten Anwendung und die Notwendigkeiten zur Eingliederung in bereits vorhandene Netzwerke. Im abschließenden Kapitel wird die Implementierung einer eigenen Applikation beschrieben.

Inhaltsverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface
AS3	ActionScript Version 3.0
DBMS	Datenbankmanagementsystem
FK	Foreign Key
GUI	Graphical User Interface
HDTV	High Definition Television
JDBC	Java Database Connectivity
LAN	Local Area Network
MAM	Media Asset Management
ODBC	Open Database Connectivity
PHP	Hypertext Preprocessor
RDBMS	relationales Datenbankmanagementsystem
RM	Rightsmanagement
SQL	Structured Query Language
SWF	Shockwave Flash
WLAN	Wireless Local Area Network

1 Einleitung

1.1 Problemstellung

In modernen Medienproduktionsunternehmen wird es unter anderem auch im Zuge der schwierigen wirtschaftlichen Verhältnisse immer wichtiger kosteneffizient und flexibel zu produzieren. Hierbei ist eine sogenannte Postproduction-Pipeline unerlässlich. Unter diesem Begriff versteht man ein softwaregestütztes System, das einen Großteil des Kommunikations- und Informationsaustausches innerhalb der einzelnen Projektteams regelt und den Projektleitern stets einen Überblick über alle aktuell laufenden Prozesse ermöglicht. Die Wirksamkeit einer an den Workflow angepassten und zuverlässig laufenden Postproduction-Pipeline wird oft zu Unrecht unterschätzt. Denn je universeller und leistungsfähiger eine solche Software ist, desto effizienter ist die Kommunikation des gesamten Teams einer Produktion.

Die Hauptaufgabe einer solchen Pipeline besteht in der Automatisierung gewisser Koordinations- und Kommunikationsaufwendungen. Darüber hinaus muss jederzeit ein bidirektionaler Informationsaustausch zwischen dem Projektleiter und seinem Team möglich sein. Eine damit einhergehende Dynamik der Daten – bezogen auf die kontinuierliche Aktualisierung der Informationen durch die Benutzer des Systems – findet sich nur in speziell auf Medienproduktionen fokussierten Anwendungen. Aus diesem Grund ist eine Postproduction-Pipeline nicht mit traditionellen Projektmanagement-Tools zu vergleichen. Zu den Grundfunktionen einer solchen Pipeline gehört unter anderem, dass ein Projektmanager neue Projekte anlegen und Supervisor zum Überwachen und Betreuen ihrer Aufgabenbereiche einsetzen kann. Der Supervisor muss dann in der Lage sein, aus dem Personalpool der Firma, Mitarbeiter mit dem Erledigen von Teilaufgaben (in der Medienbranche allgemein als Job bezeichnet) zu beauftragen. Die Mitarbeiter wiederum sollten durch die Pipeline über ihren neuen Job informiert werden um mit der Arbeit beginnen können. Sobald der Artist[1] die ihm zugetragene Aufgabe erledigt hat, setzt er seinen Supervisor über das System davon in Kenntnis.

[1] ein in der Medienbranche üblicher Begriff für einen fachspezifischen Mitarbeiter, der künstlerische Aufgaben innerhalb von Medienproduktionen erledigt

Selbstverständlich lassen sich entsprechende Lösungen auch am Markt finden, doch diese sind nicht selten so teuer, dass gerade kleine bis mittelständische Unternehmen diesen Kostenaufwand scheuen.[2] Zudem bringen sie noch weitere Nachteile mit sich. An dieser Stelle sei nur beispielhaft die Inflexibilität vieler Systeme genannt. So ist es meistens nicht möglich externe Mitarbeiter via Internet in die Pipeline einzubinden. Der Grund hierfür liegt in den meist scharfen Sicherheitsauflagen, die den Medienunternehmen auferlegt werden. Deshalb dürfen die Workstations häufig gar nicht mit dem Internet verbunden sein, auch sämtliche Schnittstellen an den Computern sind deaktiviert. Diese Vorkehrungen finden aber in Europa und generell kleinen Unternehmen nur selten Anwendung. Weil die Kunden hierzulande oftmals zudem sehr nah am Projekt beteiligt sind, ist eine Applikation die sich dahingehend öffnet durchaus erwünscht.

Der Autor hatte in einem früheren Praktikum als 3D-Artist bereits die Möglichkeit, Eindrücke hinsichtlich einer solchen Postproduction-Pipeline zu sammeln. Damals ließ sich erkennen, dass die künstlerischen Fähigkeiten der Mitarbeiter priorisiert wurden und die Organisation im Medienunternehmen zu kurz kam, worunter die Kommunikation zwischen den Mitarbeitern litt. Der Autor möchte mit Hilfe dieser Vorkenntnisse und dem Wunsch, seine Fähigkeiten als 3D-Artist mit denen eines Programmierers zu kombinieren, ein softwaregestütztes System implementieren, um die oben angesprochenen Missstände zu beseitigen.

1.2 Zielsetzung

Das Ziel des Autors ist die Entwicklung einer zukunftsorientierten Postproduction-Pipeline, die auf die Bedürfnisse eines mittelständischen Medienunternehmens zugeschnitten ist. Ein solches Unternehmen stellt die morro images GmbH & Co. KG dar, deren Tätigkeitsfelder die Erstellung visueller Effekte für Werbung, Fernsehen und

[2] Alienbrain kostet als Volumenlizenz für 25 Workstations 32.000,- $, Stand 05/2010, aktuelle Preisliste unter: http://alienbrain.com/pricing, 26.05.2010

das Kino sind. Das neue, softwaregestützte System soll eine Optimierung der zuvor dargelegten Arbeitsprozesse in diesem Unternehmen bewirken. Darüber hinaus ist die Anbindung an das Internet essentieller Bestandteil dieser Arbeit. Auf diese Weise sollen auch externe Mitarbeiter mittels Autorisierung die Möglichkeit haben, die Pipeline einzusehen und zu bedienen. Das ermöglicht einerseits ein standort-unabhängiges Arbeiten an einem Projekt. Andererseits erlaubt es, den Kunden – z.B. ein Partnerunternehmen - in die Produktionsprozesse zu involvieren und Kunden-wünsche frühzeitig abzunehmen.

Weitere Teilziele sind das Erreichen einer größtmöglichen Plattformunabhängigkeit, sowie eine Möglichkeit zur Medienintegration zu schaffen. Erstgenanntes wird benötigt, weil in Unternehmen der Medienbranche oftmals viele verschiedene Rechnersysteme verwendet werden: Die 3D-Artists benutzen in der Regel PCs auf Basis von Windows und Grafiker arbeiten gern mit MacOS. Die Administratoren hingegen verwenden ein für ihre Zwecke oftmals geeignetes Linux-System. Dies sind Betriebs-systeme an denen die Mitarbeiter tagtäglich arbeiten, wodurch es von essentieller Bedeutung ist, dass die Software auf all diesen Computern lauffähig ist. Die Integration projektspezifischer Mediendaten soll die Kommunikation zwischen den Mitarbeitern und ihren Projektleitern wesentlich vereinfachen. So soll es möglich sein Bild- bzw. Filminhalte unter anderem auch online zu diskutieren. Abnahmen könnten auf diese Weise sehr viel schneller abgewickelt werden.

Zur Umsetzung dieser Ziele werden zunächst die nötigen Anforderungen an ein solches branchenspezifisches Managementsystem erörtert. Im weiteren Verlauf dieser Erörterung werden einige Systeme, die bereits am Markt verfügbar sind, hinsichtlich ihrer grundlegenden Eigenschaften, wie Usability sowie Art und Weise der Darstellung von Informationen, untersucht. Sofern sich hieraus verwertbare Erkenntnisse ergeben, sollen diese in die Entwicklung der eigenen Anwendung mit einfließen. Darauf folgt die Analyse der bereits im Unternehmen, in Form von Excel-Tabellen, bestehenden Pipeline. Anhand dieser Untersuchung, sollen konkrete Anforderungen für die Implementierung einer zukunftsorientierten Postproduction-Pipeline erarbeitet werden. Im letzten Kapitel wird schließlich die Umsetzung einer eigenen Anwendung,

anhand von einigen ausgewählten und für die Applikation essentiell benötigten Funktionen, dargelegt.

Auf eine nach wissenschaftlichen Kriterien durchgeführte Testphase wird, aufgrund des Umfangs einer solchen Methodik und den speziellen Gegebenheiten im Unternehmen, verzichtet. Schließlich kann die Anwendung erst hinreichend getestet werden, wenn deren Entwicklung komplett abgeschlossen ist. Andernfalls könnte eine solche Testphase die laufenden Arbeitsprozesse innerhalb des Unternehmens mitunter stark behindern. Daher erfolgt eine Ergebnisbetrachtung hauptsächlich in Hinblick auf einen, den Anforderungen entsprechenden, funktionsfähigen Anwendungsprototyp.

2 Branchenspezifische Managementsysteme

2.1 Allgemeine Anforderungen

Für die Marktfähigkeit heutiger Unternehmen ist es unerlässlich, eine unterstützende Plattform in der Verwaltung einzusetzen. Der Bereich der Medienproduktion ist insbesondere für seine organisatorische Vielschichtigkeit bekannt. Straffe Vorgaben in Budget, Personalaufwand und zeitlicher Umsetzung fordern von Mitarbeitern in den Bereichen Organisation und Management ein extrem hohes Maß an Übersicht, Flexibilität, Kommunikationsfähigkeit und Belastbarkeit. Selbst kleinste Fehler in der Planungsphase können schwerwiegende Folgen nach sich ziehen. Diese offenbaren sich oftmals erst später in der Produktion und können dadurch die zeitliche Einhaltung der Fertigstellung eines Projekts gefährden. Es ist daher unumgänglich solche Risiken auf ein absolutes Minimum zu reduzieren und gleichzeitig nicht durch einen überproportionalen personellen Aufwand an Konkurrenzfähigkeit einzubüßen. Dazu bedarf es neben einem qualifizierten Planungs- und Betreuungsteam auch ein auf die individuellen Anforderungen des Unternehmens zugeschnittenes Managementsystem. Dieses muss in übersichtlicher Art und Weise alle Informationen schnell und ständig aktuell widerspiegeln können. Aber es muss auch in der Lage sein, komplexe Aufgaben in den Bereichen Projektplanung und Umsetzung zu meistern, denn die Arbeitsabläufe sind präzise zu definieren und einzuhalten. Darüber hinaus benötigt ein Medien-produktionsunternehmen, welches je nach Auftragslage kurzfristig die Mitarbeiter-anzahl deutlich verändert, eine Pipeline, welche dementsprechend die Möglichkeit bietet, flexibel auf veränderte Situationen zu reagieren.

Spätestens mit dem Beginn der Postproduction gilt es Unmengen von Daten zu strukturieren, zu verwalten und darzustellen. „Wer einmal erfolglos versucht hat, sich in der Endphase eines wichtigen Projekts durch den Wildwuchs eines verästelten Dateibaums zu kämpfen, wird das nächste Projekt verwaltungstechnisch auf eine solidere Basis stellen wollen“.[3] Wie bereits erwähnt kann sich die Mitarbeiterzahl im Laufe einer Produktion häufig vervielfachen. Daher muss sich jeder neue Mitarbeiter

[3] Natschew, Claudia: "Media Asset Management: Ein Überblick", in: Digital Production, Ausgabe 03/2004, Seite 107

ohne langwierige Einarbeitungszeit inmitten des Datenbergs zurechtfinden können. Die Ansprüche in diesem Bereich steigen nicht zuletzt aufgrund der rasanten technischen Weiterentwicklung stetig an. Trends wie HDTV und ein daraus resultierender hoher Grad an Detailreichtum, lassen das Datenaufkommen immer weiter anwachsen. Diese Erhöhung des Datenaufkommens ist auf den Einsatz von immer mehr Objekten innerhalb einer Szene zurückzuführen. Eine solche Vorgehensweise ist notwendig, um die immer klareren und schärferen Bilder auch heute noch zum Leben erwecken zu können. Darüber hinaus ist während des Aufgabenbereichs rund um das Compositing, der Einsatz von weit mehr Layern erforderlich, als dies vor einigen Jahren noch der Fall war. Mit diesen Anforderungen einher, geht die Notwendigkeit nach einer Kontrolllösung für die Verwaltung des Datenbestands.

2.2 Welche Lösungen sind am Markt vertreten?

Es befindet sich derzeit eine Vielzahl unterschiedlichster Systeme auf dem Markt, die sich der – im vorherigen Kapitel dargelegten – Anforderungen anzunehmen versuchen. Auf den ersten Blick scheint die quantitative Palette der angebotenen Produktlösungen vielfältig und den Markt hinreichend abzudecken. Es fällt jedoch bei genauerer Betrachtung die Tatsache auf, dass der technische Ansatz fast ausschließlich auf zwei grundlegend zu unterscheidenden Verarbeitungsprinzipien beruht:

Project Management verwendet das Prinzip der Organisation. Dem für die Administration zuständigen Anwender wird es ermöglicht, einen organisatorischen Rahmen für Projekte, deren Inhalte und Benutzerkonten zu erstellen und diese zu überwachen. Die Personalmanager können mittels eines automatisch mitlaufenden Kalenders einsehen wie lange ein Mitarbeiter noch mit seiner derzeitigen Aufgabe beschäftigt ist und wann er wieder frei wird. Die Verwaltung der einzelnen Projektdateien obliegt den zugewiesenen Mitarbeitern. Eine verbindliche Namensdefinition der Arbeitsdaten ist von Haus aus nicht gegeben. Für den Administrator ist die Einhaltung firmeninterner Konventionen nicht ohne weiteres überprüfbar. Der

Fokus liegt auf der Gesamtübersicht eines Projekts und der damit verbundenen Möglichkeit die Arbeitsabläufe zu jeder Zeit überwachen zu können.

Media Asset Management (MAM) verwendet das Prinzip der Verwaltung. Die Software unterstützt den einzelnen Mitarbeiter bei der Verwaltung seiner Projektdateien, beispielsweise in Form fest vorgegebener Namensdefinitionen und Verwaltungsstrukturen. Durch die programmgesteuerte Zuweisung der Daten können ausschließlich projektbezogene Mitarbeiter entsprechende Dateien einsehen oder bearbeiten. Da sämtliche Dokumente zentral auf einem Datenserver gespeichert sind wird sichergestellt, dass die Benutzer immer mit dem aktuellsten Asset[4] arbeiten. Mit Hilfe einer automatischen Versionierung aller Projektdaten wird es den Mitarbeitern jeder Hierarchieebene ermöglicht, den Fortschritt einzelner Dateien genau zu verfolgen, zu überprüfen und nachzuvollziehen. Klarer Schwerpunkt dieses Prinzips ist die dateibasierte Kontrolle, zu deren Gunst meistens auf eine umfassende Projektübersicht verzichtet wird.

2.3 Prinzip der Organisation

Als Beispiel für dieses Prinzip wird die Software *AceProject* herangezogen.[5] Sie vereint die meisten Funktionen vergleichbarer Produkte und bietet als Project Management System für Medienproduktionen die umfangreichsten Möglichkeiten der Anbindung.[6] Darüber hinaus stellt der Hersteller eine kostenlose und über einen Webbrowser einsichtige Demoversion zur Verfügung. Wie auch alle anderen Project Management Anwendungen verwendet es tabellarische Ansichten zur Darstellung projektspezifischer Daten. Diese Form ermöglicht dem Benutzer sich vielfältigste Informationen, wie zu erledigende Aufgaben, Termine, Fortschritte und Projektmitarbeiter, auf geringstem Raum anzuzeigen. Für die Projektleiter ist dies eine

[4] Besonderheit, Ergänzung, Zusatz zu einem Multimediaprodukt
[5] Homepage des Herstellers: http://www.aceproject.com/, 26.05.2010
[6] dies wurde in einem persönlichen Gespräch mit Georg Sebastian Dressler (Supervisor bei morro images GmbH & Co. KG) in Erfahrung gebracht, 17.05.2010

elegante Möglichkeit, eine umfassende Übersicht über ihr Projekt zu erhalten. Die meisten Mitarbeiter benötigen aber nur einen Bruchteil dieser Daten und – sofern sie die Systemstruktur nicht verinnerlicht haben – können sie sich in der angehäuften Informationsflut verlieren (siehe Abb. 1).

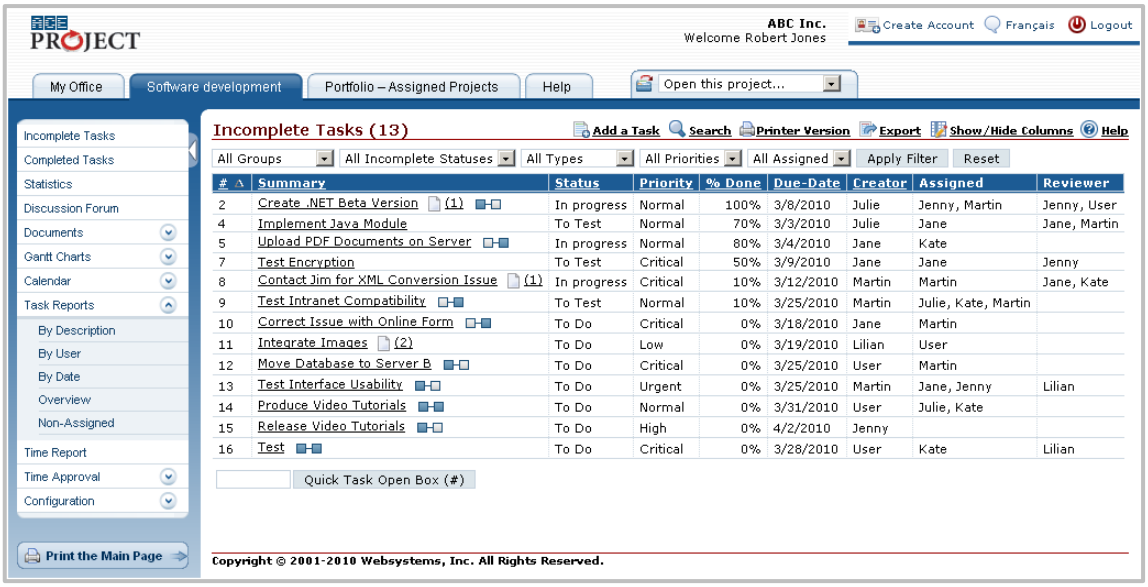


Abb. 1) tabellarische Übersicht in AceProject[7]

Solange es dem Benutzer, wie im Beispiel der Anwendung AceProject, möglich ist gezielt Informationen auszublenden, hält sich der Effizienzverlust allerdings in Grenzen. Tabellarische Strukturen eignen sich besonders zum sortierten Ausgeben der Daten. So können die Benutzer je nach persönlichen Bedürfnissen via Klick auf den Tabellenkopf eine eigene Sortierung vornehmen. Dies ist überaus hilfreich und erlaubt ein effektives Suchen nach den gewünschten Informationen. Das Filtern nach bestimmten Kriterien wiederum hilft beispielsweise dem Projektleiter beim schnellen Auffinden von Engpässen in der Produktion. In Folge dessen kann diesen kritischen Arbeitsprozessen zeitnah und zügig entgegengesteuert werden. Obwohl jeder Benutzer sich zunächst autorisieren muss um an diese Information zu gelangen, ist es, zumindest bei AceProject, nicht möglich all diese Einstellungen dauerhaft zu speichern. Nach jeder Anmeldung müssen sämtliche Eingaben erneut vorgenommen werden.

[7] eigene Darstellung, Ausschnitt aus der browserbasierten Demoversion von AceProject, https://demo.aceproject.com/Login.asp?lang=EN, 29.05.2010

Wenn der Benutzer nun eine spezifische Aufgabe näher betrachten möchte, wird das tabellarische Modul ausgeblendet und es erscheint eine detaillierte Ansicht des Jobs (siehe Abb. 2). Es ist nicht möglich beide Informationen gleichzeitig einzublenden, stattdessen ist man nun gezwungen mittels zweier Schaltflächen zwischen den Aufgaben zu wechseln. In dieser Ansicht können sämtliche Information zu dem spezifischen Job eingesehen und vom Administrator bei Bedarf geändert werden. In der sogenannten History werden alle Änderungen aufgezeichnet um den Projektleitern größtmögliche Transparenz zu gewähren. Zudem können hier auch projekttypische Dokumente an den Job angehängt werden, welche dann auf dem Server gespeichert werden.

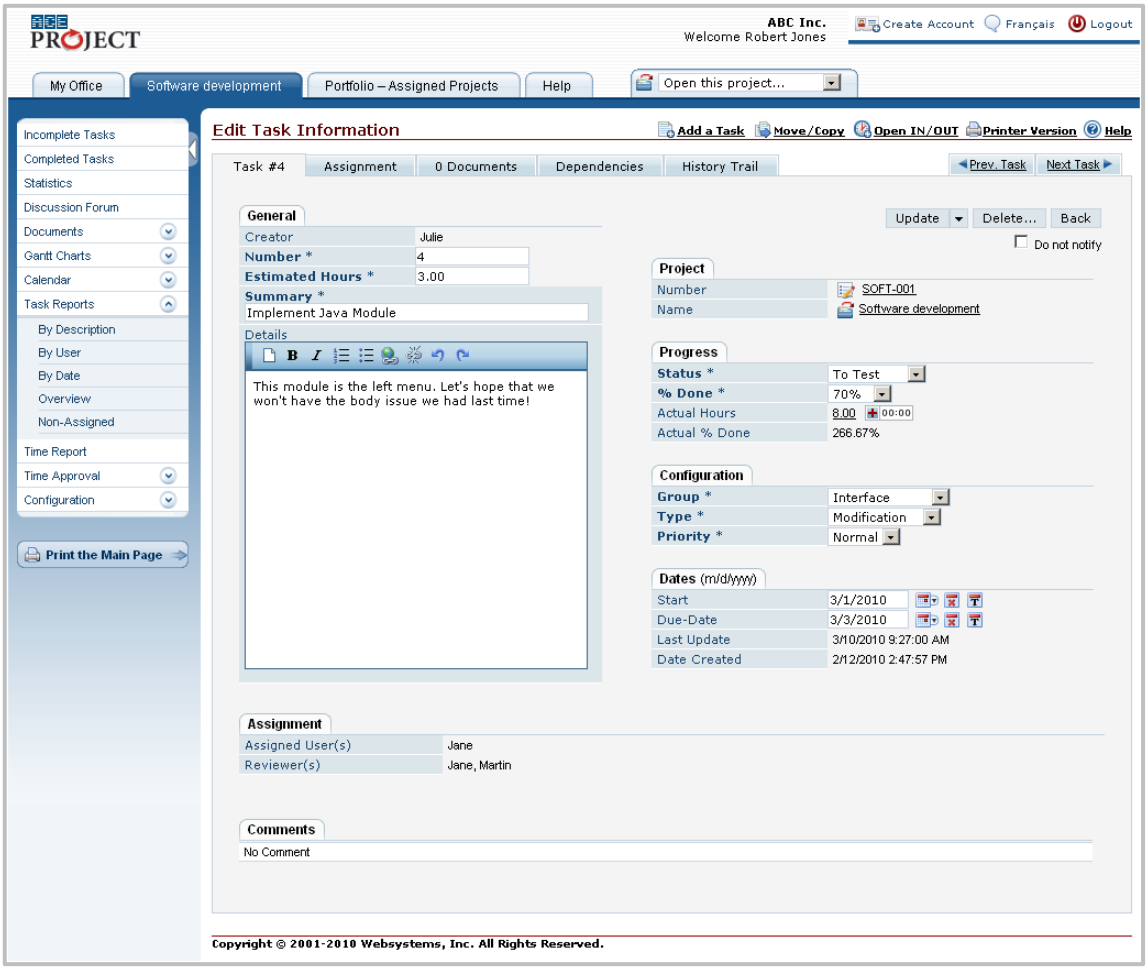


Abb. 2) detaillierte Ansicht eines Jobs in AceProject[8]

[8] eigene Darstellung, Ausschnitt aus der browserbasierten Demoversion von AceProject, https://demo.aceproject.com/Login.asp?lang=EN, 29.05.2010

Ein Project Management System präsentiert sich in der Bedienung recht statisch. Es wird dem Nutzer zwar ermöglicht Unmengen an Informationen einzublenden, doch es fehlt oftmals die Verknüpfung zwischen diesen. Möchte der Administrator dem entsprechenden Projekt beispielsweise einen neuen Mitarbeiter zuweisen, von dem er nicht genau weiß ob und wann dieser verfügbar ist, so ist er gezwungen, die aktuelle Ansicht zu verlassen um sich im Kalender darüber zu informieren. Über sogenannte Reports lassen sich übersichtliche Listen erstellen, die von den Projektleitern als Kontrolllösung genutzt werden können. Anhand dessen lässt sich beispielsweise ablesen mit welchen Jobs bestimmte Mitarbeiter zurzeit beschäftigt sind oder ob es Aufgaben gibt, die noch nicht zugewiesen sind. Sich automatisch aktualisierende Gantt-Diagramme unterstützen das Organisationsteam in der Konzeptions- und Entwicklungsphase (siehe Abb. 3). So können kritische Arbeitspakete schon früh erkannt und eventuell notwendige Gegenmaßnahmen bereits vor dem eigentlichen Projektbeginn in die Wege geleitet werden.

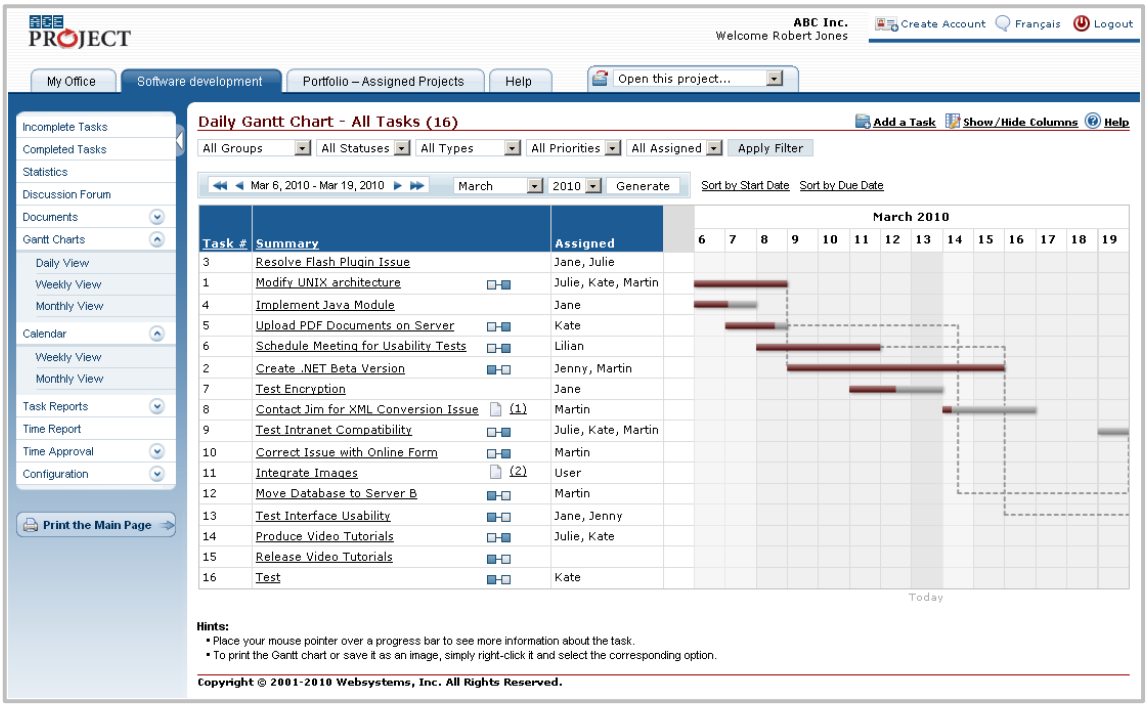


Abb. 3) Verwendung von Gantt-Diagrammen in AceProject[9]

Viele Project Management Anwendungen sind browserbasiert und erreichen dadurch eine hohe Plattformunabhängigkeit, sofern die Vielzahl der vorhanden Browser

[9] eigene Darstellung, Ausschnitt aus der browserbasierten Demoversion von AceProject, https://demo.aceproject.com/Login.asp?lang=EN, 29.05.2010

unterstützt wird. Die Managementsysteme sind in der Regel sehr ressourcenschonend und auf das Wesentliche reduziert. Auf ein dateibasiertes Interface zum Betriebssystem wird aus dieser Überlegung heraus verzichtet. Denn auch wenn die Möglichkeit gegeben ist Dokumente an die einzelnen Aufgaben anzuheften, geschieht dies immer manuell durch den Benutzer. Es existiert nicht notwendigerweise ein zentraler Speicherort für alle Mediendaten. Wie eingangs erwähnt ist es erforderlich, Namenskonventionen außerhalb des Programms zu definieren und auch zu überwachen. Dies geschieht entweder über eine systemnahe Anwendung oder mittels ausgehändigter Listen. Viele Hersteller bieten gegen monatliche Gebühren Online-Versionen ihrer Produkte an. In diesem Fall wird für den grundlegenden Betrieb eines solchen Systems ausschließlich eine Workstation benötigt. Oftmals können sie dennoch auch auf einem Server im firmeninternen LAN installiert werden.

2.4 Prinzip der Verwaltung

Zu den MAM-Systemen werden im Folgenden auch das *Document Management* und das *projektorientierte Asset Management* gezählt. All diesen Tools ist das Hauptaugenmerk – die zentrale Verwaltung von Projektdateien – gleich. Die Erörterung findet am Beispiel von Alienbrain[10] statt, weil es einerseits der Markführer der MAM-Systeme für den Bereich Feature-Film ist[11] und andererseits auch hier eine kostenlose und voll funktionsfähige Demoversion zum Testen zur Verfügung stand. MAM-Systeme werden hauptsächlich verwendet wenn große Datenmengen über einen langen Zeitraum hinweg von einem Team bearbeitet und mehrfach geändert werden. Die integrierten und automatisch kontrollierten Namenskonventionen gewinnen insbesondere dann an Bedeutung, wenn gegen Ende eines Projekts kurzfristig neue Mitarbeiter eingestellt werden müssen. Da die Mitarbeiter in einer dem Windows Explorer nachempfundenen Oberfläche arbeiten, sind sie nicht

[10] Homepage des Herstellers: http://www.alienbrain.com/, 27.05.2010
[11] dies wurde in einem persönlichen Gespräch mit Matthias Haase (CEO bei morro images GmbH & Co. KG) in Erfahrung gebracht, 17.05.2010

gezwungen sich umfangreiche Einblicke in die firmeneigene Organisationsstruktur zu verschaffen. Denn diese Struktur wird ihnen durch das Programm vorgegeben.

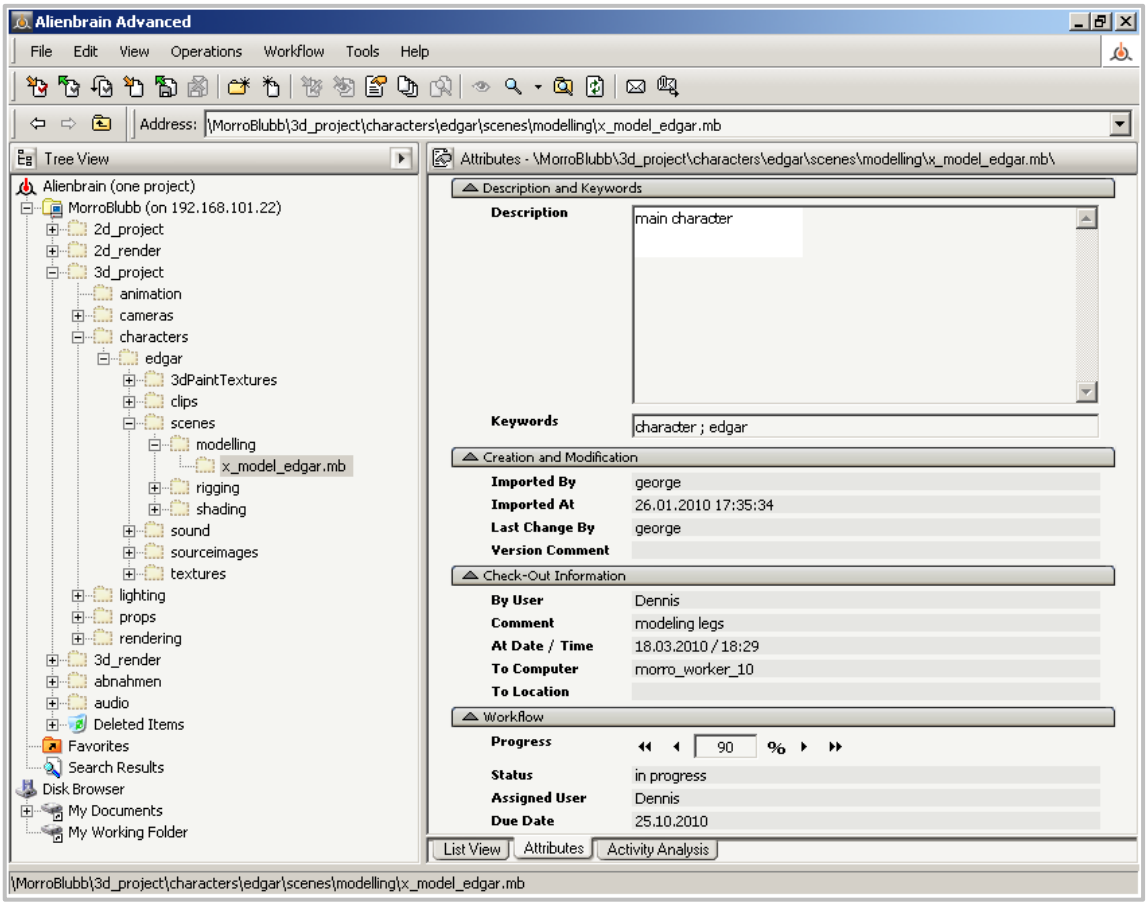


Abb. 4) dateibasiertes Interface in Alienbrain anhand eines offenen Jobs[12]

Dateien, die sich gerade in Bearbeitung befinden, werden vom System automatisch gesperrt, wodurch verhindert wird, dass mehrere Benutzer zeitgleich am selben Asset arbeiten. Die Binärdaten der Projektdateien lassen sich zudem mit Meta-Informationen anreichern. Diese werden beispielsweise genutzt um für interne und externe Abnahmen Kommentare zu unterschiedlichen Versionen einer Datei zu hinterlegen. Solche Hilfsmittel verbessern den Workflow der Künstler und die überschaubare Anzahl an Informationen garantiert den Blick auf das Wesentliche. Gerade produktionsorientierte MAM-Systeme bieten vielfältige Möglichkeiten zur Darstellung der für den Arbeitsprozess notwendigen Daten. Vergleichbar sind diese mit den typischen Ansichtsoptionen des Windows Explorers oder des Apple Finders. Vorschaubilder helfen beim Auffinden der eigenen Projektdateien. Wählt man die

[12] eigene Darstellung, Ausschnitt aus der Applikation Alienbrain

detaillierte Ansicht, so lassen sich die zuvor eingetragene Meta-Informationen einsehen (siehe Abb. 4 auf der vorherigen Seite). Diese kann man auch dazu nutzen, beim Starten externer Programme zusätzliche Parameter zu übergeben. So lässt sich beispielsweise das Interface des Programms, mit dem die Datei gestartet wird, an den Job anpassen, so dass dem Artist eine auf seine Aufgabe zugeschnittene Oberfläche präsentiert wird. Dies ist natürlich nur von Vorteil, wenn in der Firma größtenteils spezialisierte Artists engagiert sind. Sogenannte Allrounder, die verschiedenartige Aufgaben zu erledigen haben, werden sich unter Umständen an solchen Einschränkungen stören.

Das dateibasierte Interface vereinfacht die Handhabung großer Datenbäume, indem viele Ordner schon beim Erstellen der Projekte automatisch generiert werden. Die Personalmanager müssen dann lediglich noch die Beschäftigten ihren Aufgaben zuteilen, sodass jeder Artist seinen fest definierten Arbeitsbereich hat. Die Zugriffsrechte werden ebenfalls durch das System definiert. Hierdurch wird sichergestellt, dass ausschließlich befugte Mitarbeiter Dateien bei Bedarf löschen oder anderweitige Eingriffe am System vornehmen können. Es ergibt sich daraus eine große Zeitersparnis für das Organisationsteam und außerdem werden mögliche Fehlerquellen reduziert.

Die Projektleiter können, ähnlich wie bei Systemen die nach dem Prinzip der Organisation arbeiten, Berichte erstellen, die den Fortschritt der Projekte visualisieren. Ebenso ist es Personalmanagern jederzeit möglich sich einen Report zu generieren der ihnen zeigt, mit welchen Aufgaben die Mitarbeiter zurzeit beschäftigt bzw. wie viele dieser Jobs schon erledigt sind (siehe Abb. 5 auf der nachfolgenden Seite). Darüber hinaus gibt es kaum weitere integrierte Kontrollfunktionen, wodurch man gegebenenfalls auf andere Produktlösungen zurückgreifen muss. An dieser Stelle sei beispielhaft die fehlende Unterstützung seitens der Software während der Planungsphase eines Projekts genannt.

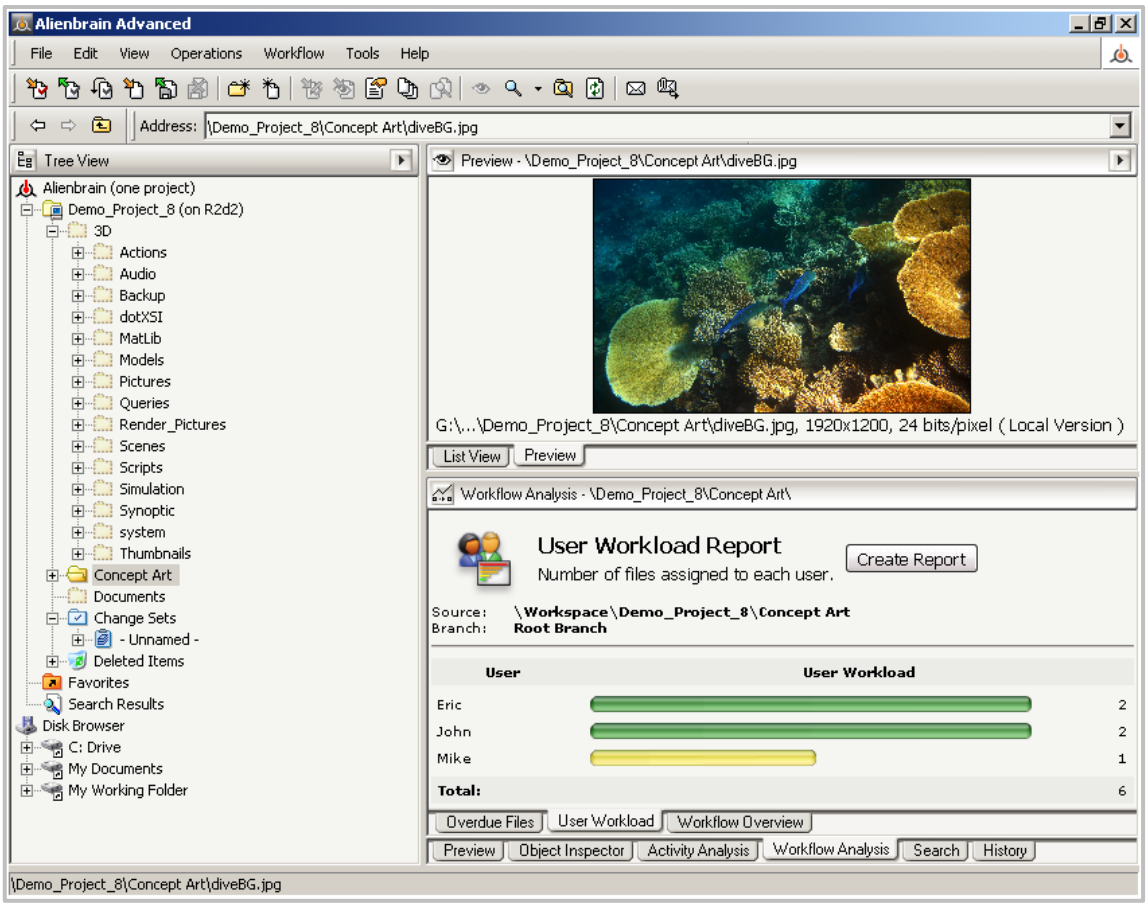


Abb. 5) weitere Ansichtsoption und Controlling in Alienbrain[13]

Aufgrund ihrer systemnahen Anbindung ist die Installation von MAM-Anwendungen sehr aufwändig und zeitintensiv. Gängige Produktionsanwendungen von Drittherstellern werden außerdem mittels eines mitgelieferten Interfaces direkt angesprochen. Daraus folgt eine gewisse Abhängigkeit vom Hersteller wenn es darum geht neue Softwareupdates bereitzustellen. Zum Betrieb wird neben dem eigentlichen Anwendungsserver im Idealfall noch ein Datenserver mitsamt Backupsystem benötigt. Zudem muss auf jedem PC eine Client-Software installiert werden. Diese stellt den zentralen Zugriff auf alle Assets sicher. In jedem Fall ist die Inbetriebnahme einer solchen Lösung um einiges kostenaufwendiger als bei einem reinen Project Management System.

[13] eigene Darstellung, Ausschnitt aus der Applikation Alienbrain

2.5 Auswertung

Während Project Management Systeme für das Organisationsteam viele Möglichkeiten für das Controlling bereit stellen, präsentieren sie sich gleichermaßen unflexibel und sind für Künstler in der Bedienung recht gewöhnungsbedürftig. Personalmanagern und Projektleitern gereicht die übersichtliche, weil geschickte, Platzierung von Unmengen an Informationen zum Vorteil. Darüber hinaus werden ihnen durch die Status-Reports und verschiedenste Diagramme mächtige Tools zur Seite gestellt, damit sie jederzeit den vollen Überblick über ihre Projekte behalten können. MAM-Anwendungen hingegen versuchen den Kommunikationsfluss innerhalb der Projektteams zu erhöhen und fordern den Mitarbeitern hinsichtlich der Bedienung wesentlich weniger ab als Project Management Systeme. Durch das direkte Arbeiten in den Systemen wird der Workflow effektiv gesteigert. Große Einarbeitungszeiten entfallen, da viele Prozesse automatisiert ablaufen, ohne dass der Künstler eingreifen muss. Durch den Wegfall vieler Controlling-Aufgaben werden auch die Projektmanager entlastet. Auf der anderen Seite gestaltet es sich schwierig, jederzeit den gesamtheitlichen Kontext zu erfassen. Viel zu oft ist es unumgänglich sich die Informationen aus dem zugegebenermaßen strukturierten Datenbestand herauszufiltern.

Bevor man sich für ein System entscheidet, sollte die Unternehmensstruktur hinsichtlich ihrer Bedürfnisse genau analysiert werden. Es gibt unzählige Möglichkeiten zum Aufbau einer firmenspezifischen Postproduction-Pipeline und genauso vielfältig gestalten sich die Lösungen. Möchte man sowohl auf eine organisatorisch wertvolle Übersicht als auch auf ein dateibasiertes Interface nicht verzichten wollen oder können, wird man nicht umhin kommen, sich mehrere Systeme anzuschaffen. Eine Alternative dazu ist die Entwicklung einer eigenen Anwendung, die im Folgenden beschrieben wird.

3 Konzeption

3.1 Anforderungsanalyse

Die Postproduction-Pipeline im untersuchten Unternehmen stellt sich bis dato durch mehrere Excel-Tabellen sowie schriftliche Anweisungen für die verschiedenen Namenskonventionen dar (siehe Abb. 6).

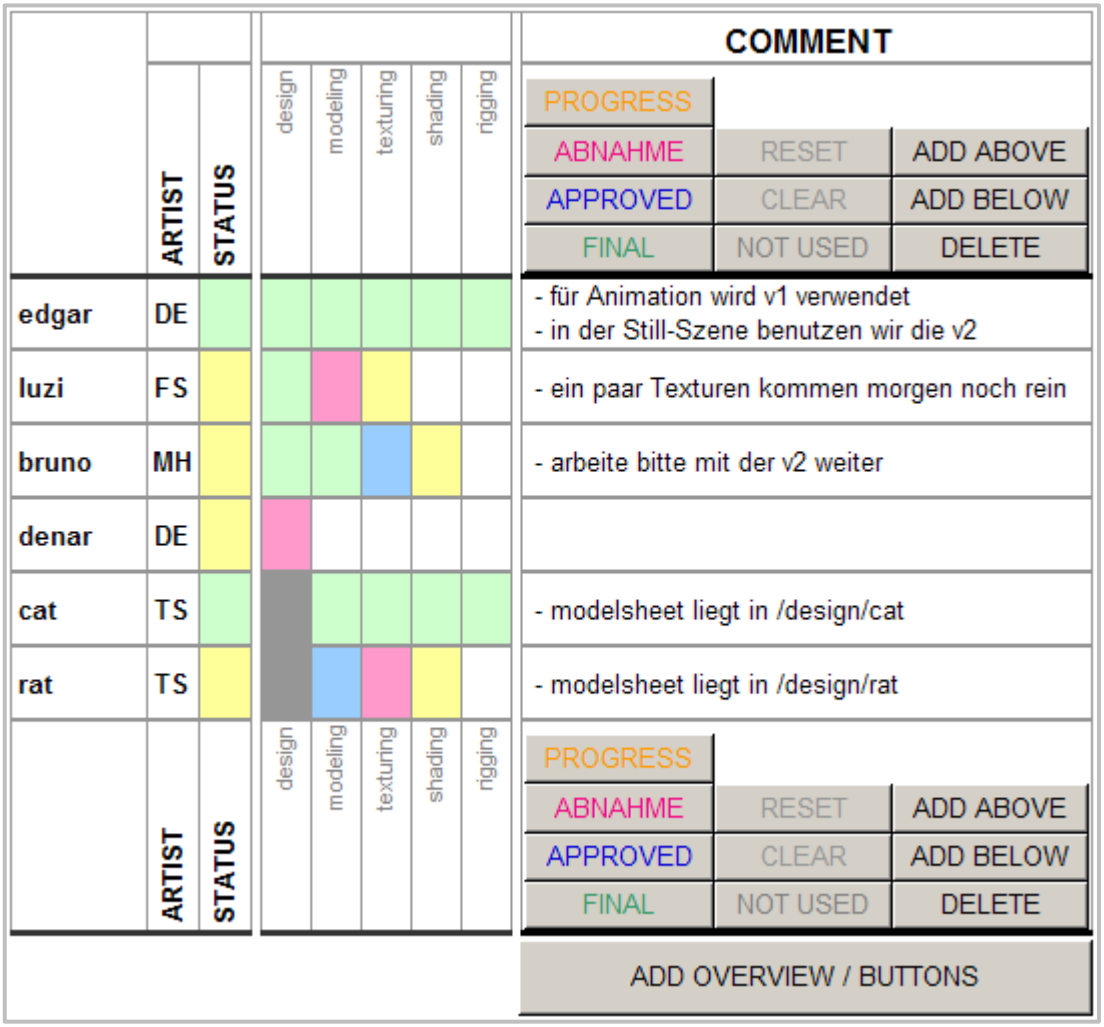


Abb. 6) Auszug einer Excel-Tabelle aus der bisherigen Pipeline[14]

In der linken Spalte wurde zunächst der Name des Objekts oder auch eines Charakters angegeben. Desweiteren ließ sich ein Kürzel für den Mitarbeiter eintragen und der Gesamtstatus festlegen. Es war nicht möglich unterschiedliche Artists für die verschie-

[14] eigene Darstellung, als Vorlage diente eine Excel-Tabelle der bisherigen Pipeline im untersuchten Unternehmen

denen Jobs eines Objekts einzutragen. Im mittleren Teil konnte für jede einzelne Aufgabe ein Status vergeben werden. Der Gesamtstatus wurde allerdings nicht automatisch aus diesen Informationen generiert, sondern musste manuell eingetragen werden. Die rechte Spalte wurde einerseits für Kommentare genutzt und andererseits wurden dort die Schaltflächen platziert, mit denen beispielsweise der Status verändert oder eine neue Zeile generiert werden konnte.

Für die bisherigen eher kleinen Aufträge konnte diese Lösung ihren Zweck zwar bedingt erfüllen. Doch waren auch hier schon erwiesenermaßen mehrere Mängel zu beobachten. Zum einen wäre hier der Zugriff auf die Excel-Tabellen zu erwähnen. Diese lagen zwar für alle zugänglich auf einem Dateiserver, konnten allerdings nicht von mehreren Clients zur gleichen Zeit verändert werden. Solange ein Mitarbeiter eine entsprechende Datei auf seinem Computer geöffnet hielt, konnten Andere sie zwar einsehen und bearbeiten, aber die Änderungen konnten nicht gespeichert werden. Ab einer bestimmten Teamgröße wird dies zwangsläufig zum Problem. Zum anderen nahm die Einarbeitung in die Pipeline viel Zeit in Anspruch. Sobald ein neuer Mitarbeiter eingestellt wurde, gab es für ihn eine umfangreiche Einweisung durch den Technischen Leiter. Diese Maßnahme nahm jedes Mal mehrere Stunden in Anspruch und da man sich mit Unmengen an Informationen konfrontiert sah, war längst nicht immer sicherzustellen, dass die neuen Mitarbeiter alles sofort verinnerlicht hatten. Einer für größere Produktionen tauglichen Anwendung müssen diese gravierenden Mängel daher als notwendige Bedingung zur Lösung gereichen. Das neue System muss die zentrale Speicherung aller projektbezogenen Informationen ermöglichen. Alle Mitarbeiter des Unternehmens müssen parallel mit dem System arbeiten können, gleichzeitig sollen ihnen ausschließlich benutzerabhängige Daten angezeigt werden. Dies wahrt nicht nur die Übersichtlichkeit, sondern vermindert zudem das Risiko an unter Umständen vertrauliche Informationen zu gelangen. Denn in Medien-unternehmen ist es – ähnlich wie bei anderweitigen Großprojekten auch – üblich, mit den Kunden Geheimhaltungserklärungen abzuschließen. Um solche Vereinbarungen nicht unnötig zu gefährden ist es von Vorteil, wenn ausschließlich am jeweiligen Projekt beteiligte Mitarbeiter auch volle Einsicht erhalten. Eine intuitive und an die

Bedürfnisse kreativer Artists angepasste Bedienung soll den Einstieg in die Software erleichtern und Meetings zur Einführung in die Pipeline deutlich verkürzen.

Weitere Probleme taten sich zudem im schlichtweg kaum existenten Controlling auf. Die Aufgaben wurden mündlich durch den Projektleiter zugeteilt und in eine der vielen Excel-Tabellen eingetragen. In der Regel wurden die Aufgaben zwar sequentiell vergeben doch es kam nicht selten vor, dass ein Mitarbeiter einen zurückliegenden Job nach einer Abnahme noch einmal zu bearbeiten hatte. Eine durchaus übliche Vorgehensweise, die den straffen Zeitvorgaben bei Medienproduktionen entgegen steht. Daraus resultierte dann eine parallele Bearbeitung mehrerer Aufgaben und damit einher ging das Risiko, dass einige Absprachen in Vergessenheit geraten konnten. Der Artist hätte in diesem Fall nur die Möglichkeit gehabt, die vorhandenen Tabellen nach seinem eigenen Namen zu durchsuchen oder mit seinem Projektleiter in Kontakt zu treten. Beide Vorgehensweisen gestalteten sich alles andere als effizient. Um dies zukünftig zu vermeiden muss eine Möglichkeit geschaffen werden, mit deren Hilfe der am System angemeldete Benutzer seine in Bearbeitung befindlichen Aufgaben jederzeit einsehen kann. Der Projektleiter sollte darüber hinaus noch im Blick behalten können welche Jobs zur Abnahme bereit stehen.

Wie bereits im Kapitel *Zielstellung* angedeutet, soll unter anderem eine erhebliche Reduzierung des Verwaltungsaufwands erreicht werden. Bisher war es unumgänglich zu jedem Projekt entsprechend neue Listen zu erstellen, die es im weiteren Verlauf zu überwachen galt. Erschwerend kam hinzu, dass für die unterschiedlichen Arbeitsabläufe während einer Nachproduktion leicht abgewandelte Tabellen zum Einsatz kamen. Gerade in den – in der Medienproduktionsbranche üblichen – kleinen Unternehmen mit vielen Generalisten[15], kam es dadurch häufig zu wechselnden Gegebenheiten, an die es sich jedes Mal anzupassen galt. Fehlende feste Regularien bezüglich der einzutragenden Informationen führten dazu, dass die Projektleiter alle getätigten Änderungen stets zu überprüfen hatten. Wurde diese Überprüfung nicht regelmäßig vorgenommen, konnte darunter schnell die Übersicht leiden und die Erfassung von nötigen Informationen wurde dadurch sehr erschwert. Deshalb bedarf

[15] Artists ohne besondere Spezialisierung, dafür können sie meist ein breites Spektrum an Aufgaben bearbeiten

es nun hinreichender Konventionen beim Eintragen neuer Informationen ohne die Mitarbeiter allzu sehr einzuschränken. Dazu zählt auch die Automatisierung stereotypischer Prozesse. Durch eine funktionsvereinigende Plattform sollen unnötige Wege im Büro und die parallele Nutzung verschiedenartiger Medientypen vermieden werden. Die anfallenden Daten müssen in einer klaren und vor allem einheitlichen Struktur abgelegt werden können. Im weiteren Verlauf der Entwicklung gilt es zu erörtern, welche Informationen grundlegend benötigt werden und welche das System nur unnötig belasten.

Eine Hürde in Hinsicht auf das zeitliche Einsparpotenzial bei größeren Produktionen stellt die bislang noch nicht vorhandene Möglichkeit dar, Informationen über das Internet abzurufen oder bereit zu stellen. So wäre es vorstellbar, sich den Status bestimmter Arbeitspakete oder die Verfügbarkeit von Mitarbeitern online anzeigen zu lassen. Zusätzlich wäre es auch von Vorteil Daten von außerhalb des Unternehmens in die Pipeline einzubinden. Sofern ein Projektleiter sich beispielsweise auf einer Dienstfahrt befand, war er angehalten, Arbeitspakete bereits im Voraus zu planen und zuzuweisen oder alternativ einen Kollegen mit dieser Aufgabe zu betreuen. Besonders zu einer Zeit, in der sich das Internet immer weiter ausbreitet und praktisch an jedem geschäftlich relevanten Ort WLAN Access Points vorhanden sind, sollte man auf die Vorteile, die sich hieraus ergeben, nicht verzichten. Befindet sich ein Mitarbeiter in einem Studio um die Dreharbeiten bezüglich der Postproduction zu überwachen, soll es ihm ermöglicht werden, relevante Informationen – beispielsweise zum Aufbau des Sets – direkt in die Pipeline hochzuladen. Bisher mussten die gewonnen Daten im Anschluss in die Excel-Tabellen transferiert werden. Hieraus gestaltet sich einerseits zwar ein enormes Einsparpotenzial, andererseits muss die Pipeline aber auch hohen Sicherheitsstandards entsprechen. Eine solche Ansammlung mitunter streng vertraulicher Daten darf unter keinen Umständen von unbefugten Personen eingesehen werden.

Der Business Administration war es bisher nur unter großem zeitlichem Aufwand möglich zu überprüfen, ob die Produktionen gewinnbringend wirtschafteten. Während eines laufenden Projekts wurden keinerlei Informationen dazu hinterlegt, wie viel Zeit

die Mitarbeiter zum Erledigen ihrer Aufgaben benötigten. Erschwerend kam hinzu, dass unter anderem auch die Business Administration zu wenig in die Projekte involviert war. Die Kommunikation zwischen den Departments war zu statisch und unorganisiert. Absprachen wurden fast ausschließlich mündlich getroffen, es fehlte eine abteilungsübergreifende Plattform. An diesem Punkt soll das neue System ansetzen. Den Projekt- und Finanzplanern sollen Werkzeuge zur Verfügung gestellt werden, mit denen sich zukünftige Projekte effektiver organisieren lassen. Dabei ist stets darauf zu achten, dass die integrierten Features keinen Mehraufwand für andere Mitarbeiter bedeuten. Schon einfache Funktionen für das Controlling würden dem Unternehmen helfen effizienter zu arbeiten, weil sie dem Organisationsteam schnell die eventuell vorkommenden Schwachstellen innerhalb der Projekte aufzeigen können. Zu diesem Zweck soll es die Möglichkeit geben sich anzeigen zu lassen, mit welchen Aufgaben die Mitarbeiter zurzeit beschäftigt sind. Das Unternehmen soll auf diese Weise näher zusammenrücken und kann dadurch an Konkurrenzfähigkeit gewinnen.

Der angestrebte modulare Aufbau der Anwendung soll einerseits das spätere Hinzufügen von weiterem Content erleichtern. Andererseits ist es auch bei der Implementierung der bereits geplanten Funktionen hilfreich. Beispielsweise werden das Debuggen[16] des Programmcodes und die damit zwangsläufig einhergehende Fehlersuche vereinfacht. Wenn sich die Funktionen auf mehrere Module verteilen, lassen sich Fehler im Programm oftmals wesentlich schneller lokalisieren. Wird der Fokus schon von Beginn an auf einen modularen Aufbau hin ausgerichtet, lassen sich auch kurzfristig neue Inhalte mit geringem technischen und zeitlichen Aufwand implementieren. Daher ermöglicht der modulare Aufbau eine Anwendung zu entwerfen, die auf die derzeitigen Bedürfnisse des Unternehmens abgestimmt ist. Gleichzeitig muss auf die Möglichkeit, auf zukünftige Entwicklungen flexibel reagieren zu können, nicht verzichtet werden. Eine solche Entwicklung könnte schon das nächste größere Projekt darstellen. Viele auf dem Markt befindliche Systeme sind von solch einer Komplexität wie sie nur wenige mittelständige Unternehmen zwingend benötigen. Gegebenenfalls wird man sogar gezwungen, funktionierende und ver-

[16] bezeichnet das Diagnostizieren von Fehlern in Computersystemen

innerlichte Arbeitsabläufe neu zu definieren. Daher sollte die Entwicklung einer auf die individuellen Bedürfnisse des Betriebs ausgerichteten Applikation von besonderem Interesse für die Produktionsleitung sein.

3.2 Technologische Eckpfeiler

Zur Realisierung dieses Projekts kommen drei Kerntechnologien zum Einsatz, die im Folgenden beschrieben werden.

Ein **Datenbank-Management-System (DBMS)** ist ein Softwarepaket, welches in der Lage ist elektronische Daten zu verwalten. Es setzt sich aus einer Kollektion von Daten, die üblicherweise Datenbank genannt wird, zusammen. Darüber hinaus gehört zu einem DBMS eine Sammlung von Werkzeugen zur Erzeugung und Verwaltung der Datenbank. Die Begriffe Datenbank und DBMS haben also verschiedene Bedeutungen und sollten nicht verwechselt werden.[17] In der Regel ist allerdings beim Verwenden des wesentlich geläufigeren Begriffs Datenbank ein DBMS gemeint. Die hauptsächliche Aufgabe eines solchen Systems besteht darin, große Datenmengen effizient, widerspruchsfrei und dauerhaft zu speichern und benötigte Teilmengen für Benutzer und Anwendungsprogramme bereitzustellen. Heutzutage kommt meist das relationale Datenbankmodell nach Edgar F. Codd zum Einsatz. Dessen Grundlage ist das Prinzip der Relation und bedeutet, dass alle Informationen in Tabellen abgelegt werden können. Es begründet sich aus der Tatsache, dass die Tabelle eine einfache und anschauliche Form zur Darstellung von Daten ist. „Von jeher sind wir es gewohnt, tabellarische Datensammlungen ohne Interpretationshilfen zu lesen und zu verstehen“.[18] Das entsprechende DBMS wird auch als relationales Datenbank-Management-System (RDBMS) bezeichnet und findet als Solches in der eigens entwickelten Applikation seine Verwendung.

[17] Kleinschmidt, Peter ; Rank, Christian: "Relationale Datenbanksysteme: Eine praktische Einführung", S. 1, Heidelberg 2005
[18] Meier, Andreas: "Relationale Datenbanken: Leitfaden für die Praxis", S. 1, Heidelberg 2004

PHP ist eine serverseitige Skriptsprache zur Erstellung dynamischer Webseiten. Sein Erfinder Rasmus Lerdorf nannte sie *Personal Home Page*. Im Laufe der Zeit wurde daraus die rekursive Version *PHP HyperText Preprocessor*. Sie eignet sich aufgrund ihres Designs ideal für Webanwendungen und ermöglicht seit Version PHP 5 auch eine objektorientierte Programmierung. Ein ausgewiesenes Ziel von PHP ist es, die Arbeit mit Datenbanken zu erleichtern.[19] Die leichte Erlernbarkeit und ihr Open Source Charakter haben maßgeblich dazu beigetragen, dass sie heute die am meisten eingesetzte serverseitige Skriptsprache ist. Eine PHP-Datei wird grundsätzlich auf einem Webserver ausgeführt. Mittlerweile wird sie von allen gängigen Plattformen wie Unix, Linux oder Windows unterstützt.

ActionScript (AS) ist eine Programmiersprache des Softwareunternehmens Adobe Systems. ActionScript-basierte Programme arbeiteten stets ereignisorientiert. Seit Version 3.0, welche nicht mehr kompatibel zu früheren Versionen ist, wird zudem auch eine klassenbasierte, objektorientierte Programmierung ermöglicht. Während ActionScript anfangs ausschließlich für Flash-Animationen verwendet wurde, findet es sich heute in immer komplexeren Webanwendungen, sogenannten Rich Internet Applications (RIA), wieder. Zur Erstellung interaktiver Inhalte wird die proprietäre Entwicklungsumgebung Flash verwendet. Es existieren zwar noch andere Frameworks, allerdings bietet sich Flash sowohl für die Programmierung einer Webbrowser-Anwendung, als auch zur Implementierung einer Desktop-Applikation besonders an. Zum einen muss der Programmcode nur einmal kompiliert werden und zum anderen sind keinerlei Anpassungen innerhalb des Codes nötig. Für das Betrachten der Inhalte in einem Webbrowser ist zwingend der Flash-Player erforderlich. Für die Desktop-Anwendung wird die Adobe Integrated Runtime (AIR) verwendet. Diese plattform-unabhängige Laufzeitumgebung dient als Container für die kompilierten Flash-Dateien im SWF-Dateiformat (Shockwave Flash). Die Erstellung einer AIR-Anwendung erfordert neben der SWF-Datei noch ein Sicherheits-Zertifikat. Darüber hinaus können noch weitere Informationen wie beispielweise die Versionsnummer und das typische Installationsverzeichnis angegeben werden.

[19] Balzert, Heide: "Basiswissen Web-Programmierung“, S. 200, Herdecke 2007

3.3 Pflichtenheft

Der Ansatz zur Lösung der beiden in der Anforderungsanalyse gestellten notwendigen Bedingungen ist eine datenbankbasierte Anwendung mit Mehrbenutzersystem. Sie verleiht dem System den geforderten zentralen Charakter. Hierdurch können beliebig viele Mitarbeiter auf die gleichen Ressourcen zugreifen. Eine solche Applikation muss in die zum Teil bereits vorhandene Netzwerkstruktur integriert werden ohne laufende Arbeitsprozesse zu behindern. Deshalb werden sowohl ein zusätzlicher Datenbank- als auch ein Webserver in die existierende Systemarchitektur eingebunden. Die Datei-Server bleiben bestehen und können auch während der Entwicklungsphase wie gewohnt verwendet werden. Durch die Verwendung neuer Rechnerkapazitäten wird zum einen der Arbeitsfluss nicht behindert und zum anderen können später umfangreiche Testszenarien durchgeführt werden.

Über den Webserver werden die eingehenden Anfragen der Clients verarbeitet, die entsprechenden Ergebnisse generiert und schließlich zurückgegeben. Hierbei ist es irrelevant, ob die Requests innerhalb des LANs oder außerhalb via Internet generiert werden. Der vorhandene Dateiserver wird verwendet, indem auf ihm ein weiterer Arbeitsbereich eingerichtet wird. Hier sollen angehängte Dokumente gespeichert werden. Auch der Backup-Server bleibt in seiner ursprünglichen Funktion bestehen. Es werden lediglich weitere Backup-Routinen hinzugefügt, die das Sichern der Datenbank und des neuen Arbeitsbereichs auf dem Datei-Server übernehmen. Die firmeninternen Computer greifen wie gewohnt per Switch bzw. Hub auf die neue Netzwerkressource zu. Externe Geräte werden via Internet vom Router auf den Webserver weitergeleitet. Dies wird mit Hilfe einer Portweiterleitung durch den Router realisiert. Nachfolgend findet sich eine grafische Übersicht zum Aufbau der Systemumgebung (siehe Abb. 7 auf der nachfolgenden Seite)

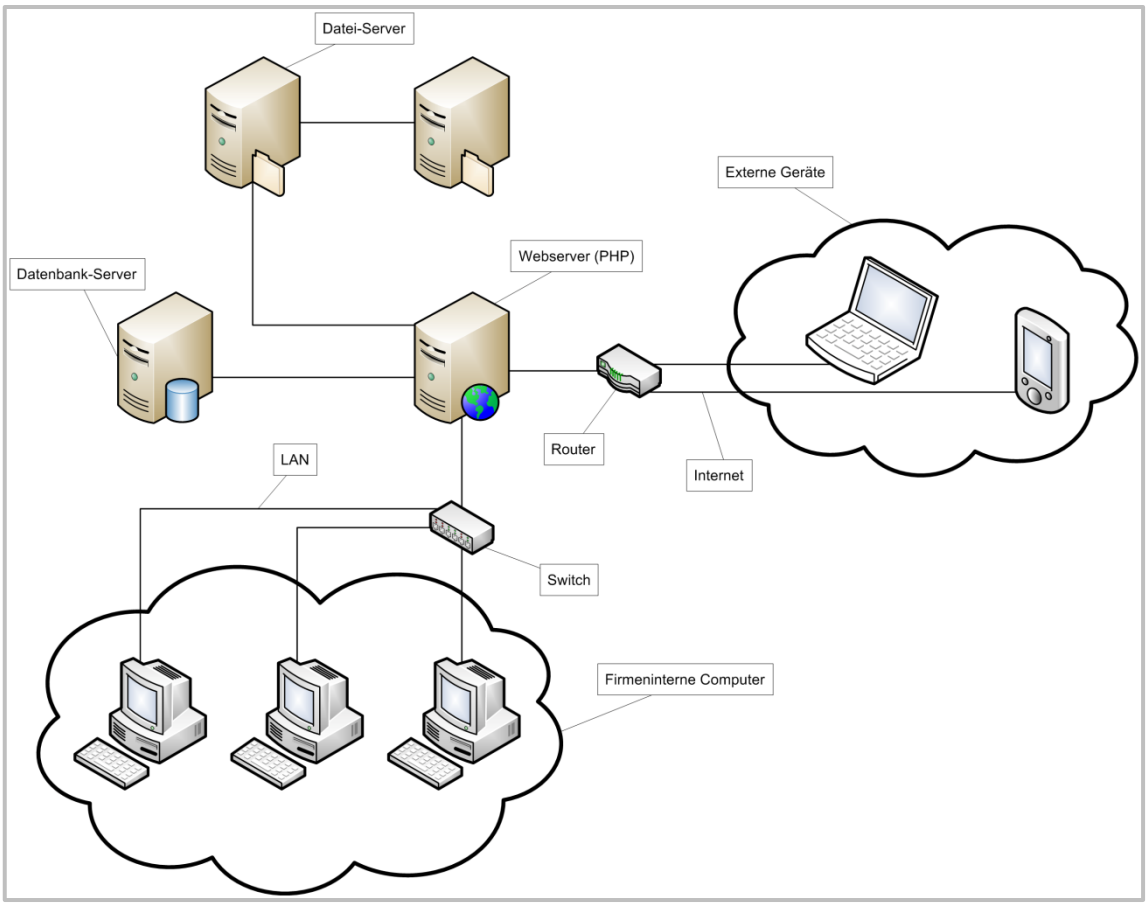


Abb. 7) Systemumgebung der Postproduction-Pipeline[20]

Es werden zwei Möglichkeiten bereitgestellt mittels derer die Verbindung zur Postproduction-Pipeline erfolgen kann. Zum einen ist dies eine Desktop-Anwendung im klassischen Sinne. Diese soll auf den firmeninternen Geräten installiert werden und gewährleisten, dass keine Abhängigkeit vom Browser besteht. Zum anderen wird man sich aber auch über eben jenen Browser mit dem System verbinden können. Diese Möglichkeit soll vorrangig bei externen Mitarbeitern und mobilen Endgeräten zum Einsatz kommen. Um den Aufwand in programmiertechnischer Hinsicht diesbezüglich so gering wie möglich zu halten, wird als Programmiersprache auf ActionScript zurückgegriffen (vgl. Kapitel 3.2). Mit Hilfe der verwendeten Flash-Laufzeitumgebung ist es zumindest derzeit möglich die Anwendung in allen gängigen Browsern aufzurufen. Dadurch wird eine solide Plattformunabhängigkeit gewährleistet. Wie bereits beschrieben, wird für die Desktop-Anwendung die Adobe Integrated Runtime verwendet. Da AIR hauptsächlich für die Entwicklung von RIAs eingesetzt wird, eignet sie sich besonders für den geplanten Anwendungszweck. Sie erweitert AS3 zudem um

[20] eigene Darstellung

zusätzliche Klassen, die zum Beispiel in der Lage sind direkt auf das Dateisystem zuzugreifen.

Da AS3 über keine direkten Schnittstellen für Datenbanken verfügt, muss eine entsprechende Anbindung auf Seiten des Webservers erfolgen. Diese Aufgabe werden PHP-Skripte übernehmen (vgl. Kapitel 3.2). Sie bekommen vom Client eine Anfrage, leiten diese zur Datenbank weiter und senden dann den jeweiligen Response zurück. Auch anfallende Requests bezüglich angehängter Dokumente können auf diese Weise verarbeitet werden. In jenem Fall erfolgt die Abfrage über den speziell eingerichteten Arbeitsbereich auf den Datei-Servern (siehe Abb. 8).

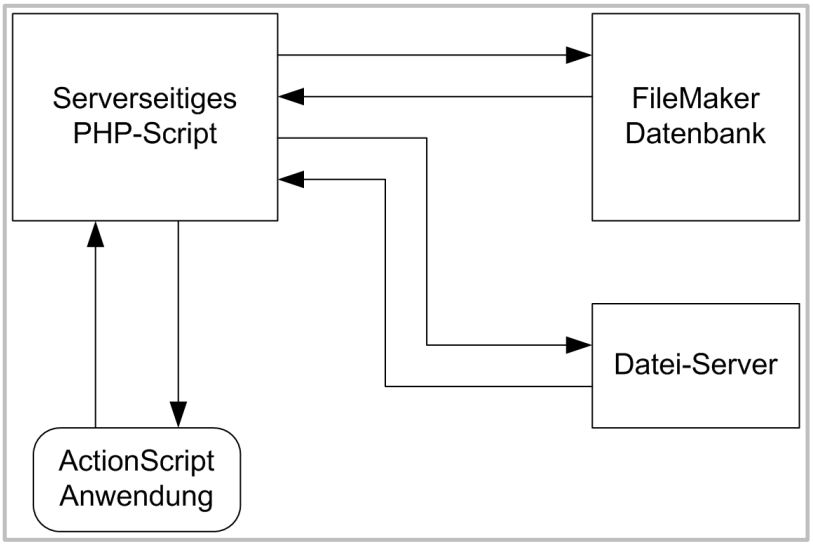


Abb. 8) Datenflussdiagramm der Postproduction-Pipeline[21]

Die Oberfläche der Anwendung wird in drei voneinander getrennte Bereiche unterteilt, in denen unterschiedliche Module untergebracht sind. Dadurch lassen sich einerseits verschiedene Informationen gleichzeitig betrachten und andererseits optimieren sie den Workflow. So benötigen die Mitarbeiter für ihre künstlerischen Tätigkeiten hauptsächlich das sogenannte JobView-Modul. Hier findet eine Auflistung aller zurzeit aktiven Aufgaben des eingeloggten Artists Platz. Diese werden nach Projekten sortiert und grafisch voneinander getrennt gruppiert. Die generierten Informationsblöcke enthalten alle relevanten Daten, wie Name und Abgabetermin, sowie den Status des Jobs. Darüber hinaus fungieren sie gleichzeitig als Schaltflächen. Über einen Klick mit

[21] eigene Darstellung

der linken Maustaste wird der ausgewählte Job mit allen detailierten Informationen im rechten oberen Modul – dem InfoView – präsentiert. Im unteren Bereich übernimmt eine tabellarische Struktur die Navigation durch alle vorhandenen Projekte und deren Untergruppen. Über das rechtsbündige Menü sind verschiedene Aktionen und Optionen auswählbar. Auch die Autorisierung beim Starten der Applikation erfolgt über dieses Modul. Die Menüleiste ist ein- bzw. ausblendbar, um bei Bedarf ein wenig Raum zu gewinnen oder nicht benötigte Eingabeoptionen zu verbergen. Am unteren Rand ist eine Legende platziert, welche eine Erklärung zu den verwendeten Statusfarben liefert. In Hinblick auf die spätere Gestaltung des Layouts, wurde bereits im Vorfeld eine grobe Previsualisierung erstellt (siehe Abb. 9).

Abb. 9) Previsualisierung für die Gestaltung des Layouts[22]

Um eine gute Usability zu erreichen, wird in höchstem Maße auf Interaktions-fähigkeiten geachtet. So dient die Tabelle des sogenannten OverView-Moduls nicht nur der Veranschaulichung der Daten, sondern kann zugleich auch zur Navigation

[22] eigene Darstellung

verwendet werden. Vergleichbar ist diese Art des Navigierens mit dem von Microsoft Windows verwendeten Dateiexplorer. Anstatt von Symbolen kommen hingegen die bereits erwähnten, tabellarisch angeordneten Zeilen zum Einsatz, weil diese zusätzlich noch weitere Informationen beherbergen können. Eine darüber platzierte Statusleiste hilft beim Navigieren und sorgt für die notwendige Übersicht. Weiterhin wurde darauf geachtet, dass sich alle für das jeweilige Modul benötigten Schaltflächen in unmittelbarer Nähe zu diesem befinden. Wenn ein Projektleiter beispielsweise die Informationen zu einer Aufgabe ändern muss, so befinden sich die entsprechenden Buttons auch innerhalb dieses Bereiches – im vorliegenden Fall direkt darunter. Diese kurzen Mauswege ermöglichen einerseits einen effizienten Workflow und sind andererseits sehr benutzerfreundlich. Benötigen Mitarbeiter die Kontaktdaten ihres Supervisors, so gelangen sie über eine kleine Schaltfläche hinter dem Namen zu seiner Seite. Hier befinden sich auch weitere Daten zu dieser Person, wie Handynummer und Adresse. Die Anwendung kann für den Personalmanager dementsprechend auch als fundamentales Mitarbeiterverzeichnis verwendet werden.

Als erste und zugleich nach innen gerichtete sicherheitsrelevante Maßnahme wird ein System für die Verwaltung des Rechtemanagements integriert. Es soll berechtigten Benutzern spezielle Einsichten oder Rechte gewähren bzw. diese verwehren. So soll den Geschäftsführern und dem Technischen Leiter voller Einblick in alle laufenden Projekte erlaubt werden, während Projektleitern alle möglichen Werkzeuge zum Verwalten zur Verfügung gestellt werden. Die Künstler wiederum haben kaum Rechte und der Einblick in Projekte, an denen sie nicht beteiligt sind, soll ihnen verwehrt bleiben. Auf diese Weise lassen sich nicht nur Geheimhaltungsvereinbarungen im Unternehmen leichter durchsetzen. Es wird auch garantiert, dass unbefugte Mitarbeiter keinerlei Möglichkeiten besitzen, Daten unbeabsichtigt zu ändern oder zu löschen.

Um sich Zugriff auf die Anwendung zu verschaffen ist es notwendig sich vorher am System zu autorisieren. Dafür wird ein Anmeldeformular benötigt, in welches der Mitarbeiter seinen Benutzernamen und sein Passwort einträgt. Dieses Szenario wurde nachfolgend grafisch dargelegt (siehe Abb. 10 auf der nachfolgenden Seite).

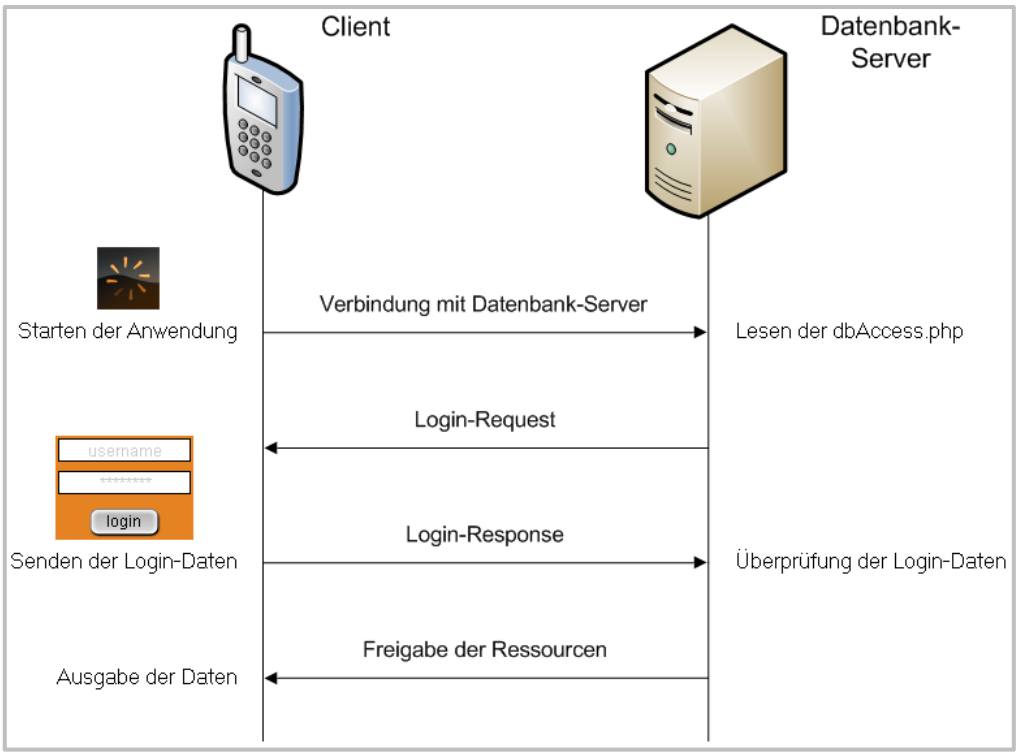


Abb. 10) Datenflussdiagramm für die Autorisierung[23]

Alle weiteren Ressourcen können ausschließlich dann eingesehen werden, wenn diese beiden Informationen durch den Webserver erfolgreich verifiziert wurden. Andernfalls wird dem User der Zugriff verweigert. Hierdurch wird eine grundlegende Sicherheit zum Schutz nach außen gewährleistet. Nun existiert noch die Möglichkeit sich direkt am Datenbank-Management-System anzumelden und so an die hinterlegten Informationen zu gelangen. Aus diesem Grund verfügt das DBMS über passwort-gesicherte, verschlüsselte Konten. Darüber hinaus ist die Anmeldung über die PHP-Engine nur mit einem eingeschränkten Konto möglich. Für die Autorisierung über den Administrator-Login ist es unumgänglich die entsprechende Client-Software im LAN auszuführen.

[23] eigene Darstellung

4 Implementierung

4.1 Datenbankmanagementsystem

4.1.1 FileMaker

Für die Verwaltung der anfallenden Daten der Postproduction-Pipeline wird FileMaker verwendet. Hierbei handelt es sich um ein proprietäres Datenbank-Management-System des gleichnamigen Herstellers.[24] Der große Unterschied zu anderen Systemen dieser Art besteht darin, dass sich mit FileMaker sogenannte Layouts erstellen lassen. Diese Layouts simulieren mittels Schaltflächen und Grafiken eine interaktive GUI (Graphical User Interface). Dazu werden die Layouts mit Hilfe von Ein- und Ausgabe-Modulen erstellt, welche sich innerhalb einer definierbaren Arbeitsfläche frei platzieren lassen. Die Module sind in dem Softwarepaket enthalten und werden beim Anlegen über ein Dialogfenster mit den Feldern der Datenbank verknüpft. Diese von FileMaker gewählte Vorgehensweise ermöglicht dem User ein direktes und annähernd ohne Wartezeiten auskommendes Arbeiten, mit der im Hintergrund laufenden Datenbank. Ursprünglich sollte auf diese Art auch die Postproduction-Pipeline realisiert werden. Die nicht vorhandene Möglichkeit zur Programmierung der Oberfläche stellte sich allerdings schnell als Hindernis dar. Hinsichtlich der Usability kam es zu vielerlei Einschränkungen, weil ausschließlich die vorgegeben Module verwendet werden konnten. Die Implementierung einiger der geforderten Funktionen war so nicht zu realisieren.

Ein weiteres Problem ergab sich aus der – von FileMaker entwickelten – Skriptsprache zur Steuerung von Datenbankbefehlen. Diese bietet, dem Verwendungszweck entsprechend, Anfängern ohne Programmierkenntnisse viele Vorteile. Beispielsweise wurde die komplette Syntax in verschiedene Sprachen übersetzt. Selbige unterscheidet sich allerdings grundlegend von den meisten heutzutage verwendeten Programmier- und Skriptsprachen, weswegen sie erfahrenen Entwicklern sehr fremdartig erscheint. Eine Möglichkeit zur Verwendung der wesentlich verbreiteteren SQL-Syntax ist zumindest innerhalb des DBMS nicht vorhanden.

[24] Homepage des Herstellers: http://www.filemaker.de/, 19.05.2010

Schlussendlich stellten sich auch die finanziellen Kosten für ein solches System als Problem heraus. Denn mit der sogenannten *FileMaker Pro* Software lassen sich nur neun Clients miteinander verbinden. Für größere Netzwerke wird somit mindestens die Serveranwendung *FileMaker Server* benötigt. Diese Applikation stellt die Datenbanken jedoch lediglich auf einem entsprechenden Server bereit. Für den Zugriff von den einzelnen Workstations aus werden jeweils separate Lizenzen von *FileMaker Pro* benötigt. Die Erstellung bzw. Verwaltung der Datenbank erfolgt über das Softwarepaket *FileMaker Pro Advanced*. Durch diese Vermarktungsstrategie des Herstellers, wird selbst der Einsatz in mittelständischen Unternehmen finanziell schnell belastend.[25]

Aufgrund der dargelegten Umstände wurde zu Beginn der Entwicklungsphase die Entscheidung gefällt, *FileMaker Server* lediglich als gewöhnlichen Datenbankserver zu verwenden und die Client-Anwendung selbst zu entwickeln.

4.1.2 Installation und Konfiguration

FileMaker lässt sich grundsätzlich sowohl unter MacOS X und Windows Server betreiben. Aufgrund des im Unternehmen bereits vorhandenen Windows Netzwerks, fiel die Wahl auf Microsofts Windows Server (2003) Betriebssystem. Diese ältere Version wird verwendet, weil das Betriebssystem bereits im Unternehmen genutzt wird und der technische Leiter mit den administrativen Vorgehensweisen vertraut ist. Eine Umstellung zu einem späteren Zeitpunkt ist jedoch ohne großen Aufwand zu bewerkstelligen. Als Datenbankserver kommt *FileMaker Server Advanced* zum Einsatz. Dieser bietet zusätzlich zu der Standardversion noch eine ODBC[26] und JDBC[27] Unterstützung. Hierbei handelt es sich um Datenbankschnittstellen, die das Auslesen

[25] FileMaker Server kostet derzeit 999,- € und jede FileMaker Pro Lizenz kostet 349,- €, Stand 05/2010, Preisliste einsehbar unter: http://filemaker.de/products/compare/, 19.05.2010

[26] Open Database Connectivity, standardisierte Datenbankschnittstelle die SQL als Datenbanksprache verwendet

[27] Java Database Connectivity, Datenbankschnittstelle der Java-Plattform die speziell auf relationale Datenbanken ausgerichtet ist

der Datensätze durch Anwendungen anderer Hersteller ermöglichen. Derzeit werden diese Schnittstellen noch nicht zwingend benötigt. Eine Auswertung der Daten über andere Programme, beispielsweise zur Finanzplanung, ist allerdings bereits vorgesehen und wurde daher berücksichtigt.

Sowohl die Administration als auch die Erstellung der Datenbankstruktur erfolgt, wie bereits angedeutet, über die Client-Software *FileMaker Pro Advanced*. Nach der Erstellung einer geeigneten Datenbank, muss diese auf dem Datenbankserver veröffentlicht und anschließend gestartet werden. Um via PHP auf die Datenbank zugreifen zu können, ist die Initialisierung der *Web Publishing Engine* innerhalb der Admin-Konsole von *FileMaker Server* erforderlich (siehe Abb. 11). Hierbei ist der Pfad der PHP-Klassenbibliothek anzugeben. FileMaker installiert daraufhin eine API, welche PHP um weitere Klassen erweitert. Mit deren Hilfe können über PHP-Skripte ein Teil der Datenbankbefehle in FileMaker ausgeführt werden. Darüber hinaus muss im DBMS noch ein Benutzerprofil für den Zugriff über die Web Publishing Engine angelegt werden.

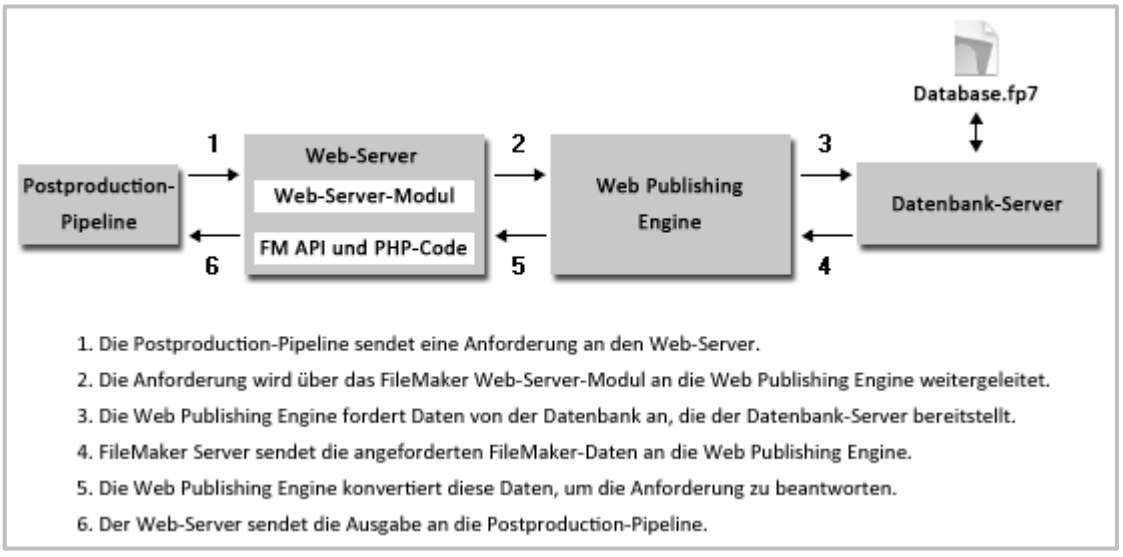


Abb. 11) Verwendung der FileMaker Server Web Publishing Engine[28]

Um eventuell auftretendem Datenverlust vorzubeugen, verfügt FileMaker über automatisierte Backup-Routinen. Diese speichern jeden Tag ein Image der gesamten

[28] eigene Darstellung, in Anlehnung an: "FileMaker 11: Custom Web Publishing mit PHP", http://filemaker.de/support/product/docs/fms/fms11_cwp_php_de.pdf, S.10f, 19.05.2010

Datenbank. Die Backups bleiben über einen Zeitraum von zehn Tagen bestehen, bevor sie überschrieben werden. Weil die Sicherungsdateien auf dem Datenbankserver abgelegt werden, wurde zusätzlich dazu ein Befehl in das bestehende Backup-System des Unternehmens integriert. So werden die Daten nun auch auf den Datei-Servern gespiegelt und vervollständigen damit die Maßnahmen gegen Datenverlust. Über die Admin-Konsole von *FileMaker Server* lassen sich die Backups durch den Administrator innerhalb weniger Minuten wieder einspielen. Mit Hilfe dieser Konsole ist es auch möglich manuelle Backups zu einem beliebigen Zeitpunkt zu erstellen. Es sollte allerdings die dabei entstehende zusätzliche Prozessorlast bedacht werden, die – in Abhängigkeit der Größe einer Datenbank – zu einer Verlangsamung des Systems führen kann.

4.1.3 Erstellung einer geeigneten Datenbankstruktur

Im Folgenden muss eine geeignete Datenbankstruktur entworfen werden. Dazu wurde die Projektstruktur typischer Medienproduktionen im untersuchten Unternehmen analysiert. Nachfolgend wurde eine hierarchische Gliederung mit konstanter Tiefe vorgenommen, auf dessen Basis die Erstellung der Tabellen vorgenommen werden konnte (siehe Abb. 12 auf der nachfolgenden Seite). Für jede Ebene innerhalb dieser Projektstruktur konnte nun eine entsprechende Tabelle konzipiert werden. Dieses Verfahren lässt sich auf nahezu alle gängigen Medienproduktionen ohne nennenswerte Einbußen in der Handhabbarkeit anwenden. Ein großer Vorteil dieser Methodik besteht in der kaum vorkommenden Datenredundanz. Diese Redundanzen gilt es nach der verbreiteten Informationstheorie weitestgehend zu vermeiden. Eine Ausnahme hiervon ist nur dann erwünscht, wenn sich die Rechenzeit der Software dadurch reduzieren lässt.

Entsprechend der dargestellten Grafik (Abb. 12 auf der nachfolgenden Seite) lassen sich auch die Beziehungen zwischen den Tabellen konzipieren. So besteht zunächst jeweils eine Beziehung zu der darüber befindlichen Ebene (mit Ausnahme der obersten

Ebene). Aus Gründen die zu einer Reduzierung des Rechenaufwands führen, wurden für jede Ebene weitere Beziehungen zu allen darüber liegenden Ebenen konzipiert.

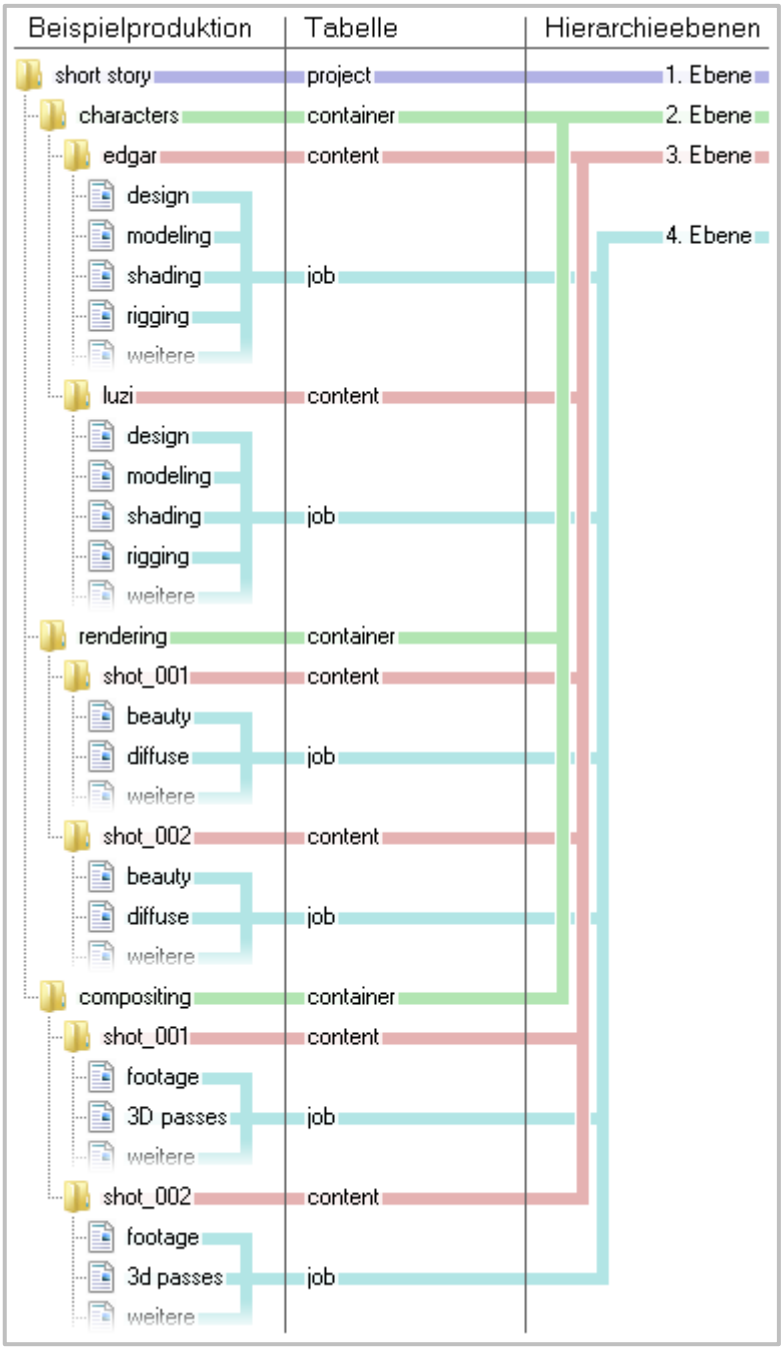


Abb. 12) Projektstruktur und Hierarchieebenen anhand eines Beispiels[29]

Aus diesen Erkenntnissen lässt sich nun eine den Anforderungen entsprechende Datenbank erstellen. Zunächst werden die vier Tabellen *Project*, *Container*, *Content* und *Job* entsprechend der vier Hierarchieebenen angelegt. Sie enthalten alle Einträge,

[29] eigene Darstellung

welche für die Projektstruktur erforderlich sind. Darüber hinaus wird eine Tabelle *Task* benötigt. Sie speichert alle Jobs, die innerhalb der *Content-Objekte* eines *Containers* erforderlich sind. Beim Erstellen eines neuen *Content-Objekts* müssen die Aufgaben somit nicht jedesmal aufs Neue eingegeben werden. Die Informationen der Tabelle *Task* sind zwar redundant, aber bei der Erstellung von Projekten hinsichtlich des Komforts und des Zeitaufwands unerlässlich. Eine sogenannte *Communicator*-Tabelle speichert alle Einträge, die innerhalb des gleichnamigen Moduls generiert werden. Diese Informationen dienen hauptsächlich dazu, Änderungen an den einzelnen Aufgaben auch über längere Zeit nachvollziehen zu können. Schlussendlich wird noch eine Tabelle – genannt *Artist* – für das Anlegen der Benutzerkonten verwendet. Hier werden sowohl Informationen zu den Usern als auch ihre gültigen Rechte und ihre Login-Daten hinterlegt (siehe Abb. 13).

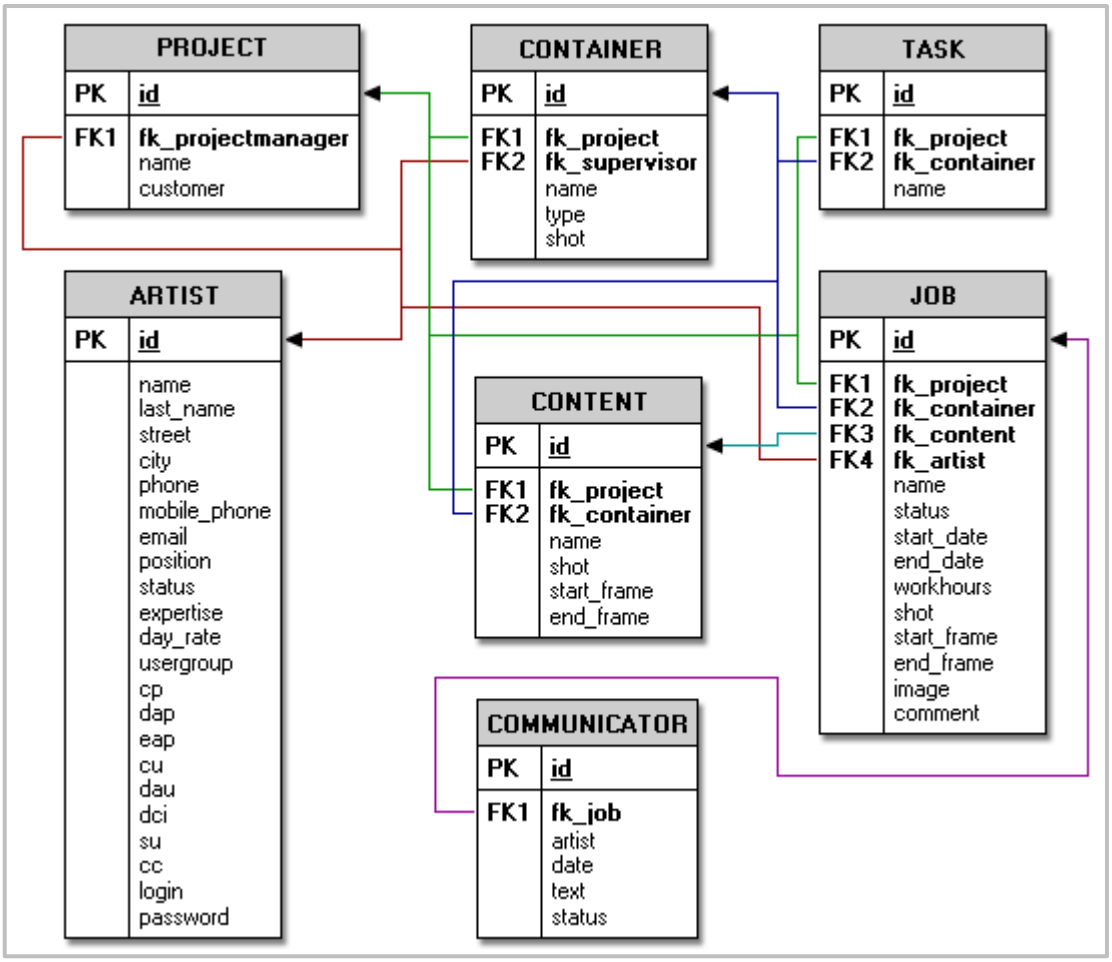


Abb. 13) Datenbankstruktur der Postproduction-Pipeline nach dem relationalen Datenbankmodell[30]

[30] eigene Darstellung, Herangehensweise beschrieben in: Moos, Alfred: "Datenbank-Engineering: Analyse, Entwurf und Implementierung objektrelationaler Datenbanken", Wiesbaden 2004

Dem Prinzip des relationalen Datenbankmodells folgend, verfügen alle Tabellen über Primärschlüssel (Primary Keys), die mit *id* benannt sind. Zudem beinhalten einige Tabellen Fremdschlüssel (Foreign Keys), welche grundsätzlich mit dem Kürzel *fk* beginnen. Die Foreign Keys stellen die Beziehungen zu den Tabellen her. Da die Nummer des Primärschlüssels innerhalb jeder Tabelle genau einmal auftritt, kann jeder Datensatz eindeutig referenziert werden. Die Beziehungen zwischen den Tabellen sind zum Teil aus den Überlegungen hinsichtlich der Projektstruktur abzuleiten. Alle weiteren Beziehungen ergeben sich aus ihrem Verwendungszweck heraus.

4.1.4 Übersicht über die Tabellen

Im Folgenden wird der Aufbau der einzelnen Tabellen analysiert und die Funktion der verwendeten Einträge erläutert.

Project-Tabelle:

PROJECT

id	fk_projectmanager	name	customer
1	2	productionPipeline	morro images
2	2	moffels	RBB
3	6	brunoBanani	Dorland Grey
4	8	poe	morro images
5	6	schwarzatmen	morro images

Abb. 14) Ausschnitt aus der Tabelle PROJECT[31]

In dieser Tabelle werden die Projektordner abgelegt, entsprechend wird für jeden Auftrag ein neuer Eintrag generiert. Über den Fremdschlüssel *fk_projectmanager* wird mit jedem Eintrag genau ein Projektleiter aus der Artist-Tabelle verknüpft. Zunächst wird hier automatisch derjenige Benutzer eingetragen, über dessen Benutzerkonto das Projekt erstellt wurde. Im Nachhinein ist es möglich jeden Mitarbeiter, der über das entsprechende Recht verfügt, einzutragen. Der Projektleiter besitzt innerhalb seines

[31] eigene Darstellung

Projekts volle Rechte. Er kann neue Inhalte anlegen und diese wiederum verändern oder löschen. Wechselt der Projektleiter während der Produktionsphase, gehen alle seine Rechte an den Nachfolger über. Es kann immer nur ein einziger Projektleiter eingetragen werden. Der Eintrag *name* hilft beim identifizieren des Projekts und wird in den Tabellenzeilen der Applikation angezeigt. In *customer* wird der Name des Auftraggebers festgehalten. Diese Information wird ebenfalls in der jeweiligen Tabellenzeile eingeblendet. Um die Suche nach einem bestimmten Projekt zu erleichtern, findet mit der Ausgabe eine Sortierung nach der Spalte *name* statt.

Container-Tabelle:

CONTAINER

id	fk_project	fk_supervisor	name	type	shot
1	2	2	characters	object	false
2	2	2	props	object	false
3	2	2	sets	set	false
4	4	4	animation	shot	true
5	4	4	rendering	rendering	true

Abb. 15) Ausschnitt aus der Tabelle CONTAINER[32]

Diese Tabelle enthält wiederum Ordner, in denen zueinander passende Objekte gruppiert werden. So lassen sich beispielsweise die vorhandenen Charaktere eines Films in einem gemeinsamen Container *characters* unterbringen.[33] Die Spalte *fk_project* enthält den Fremdschlüssel um die Beziehung zum entsprechenden Projekt herstellen zu können. Darüber hinaus können den Containern jeweils ein Supervisor zugewiesen werden. Sein Primärschlüssel wird analog zum Projektleiter in der Project-Tabelle unter *fk_supervisor* eingetragen. Im Gegensatz zu diesem, muss der Supervisor hingegen auch beim Erstellen manuell angegeben werden, weil er seine eigenen Container nicht eigenständig anlegen kann. Innerhalb seines Aufgabenbereichs besitzt der Supervisor dieselben Rechte wie der Projektleiter. Unter *name* wird die Bezeichnung des Containers abgelegt und auch hier wird dieser Eintrag für die Tabellenzeile verwendet. Der Eintrag *type* bestimmt die Art des Containers und legt fest, welche vorgegebenen Aufgaben die enthaltenen Objekte zunächst besitzen.

[32] eigene Darstellung
[33] vgl. Abb. 12 auf Seite 36

Dabei wird zwischen folgenden Typen und ihren Aufgaben unterschieden:

Name	Object	Set	Shot	Compositing	Rendering
Shot	false	false	true	true	true
Aufgaben	design modelsheet modeling shading rigging blendshape	design modelsheet modeling shading rigging mattepainting lighting	staging blocking animation simulation baking camera tracking lighting	footage 3D passes tracking keying rotoscoping color correction final rendering	beauty diffuse spec ao aoGround shadow shadowGround dof reflection movec light mask bg

Abb. 16) verwendete Containertypen und ihre vordefinierten Aufgaben[34]

Auf diese Weise lassen sich ohne großen Zeitaufwand gängige Projektstrukturen erzeugen.[35] Da die vordefinierten Jobs mit Hilfe von Checkboxen dargestellt werden, lassen sie sich bei Bedarf beliebig an- oder abwählen. Darüber hinaus können weitere Aufgaben manuell über ein Texteingabefeld hinzugefügt werden. Alle ausgewählten und eventuell ergänzten Aufgaben werden in der Task-Tabelle für eine Weiterverarbeitung zwischengespeichert. Die Spalte *shot* kann entsprechend des booleschen Datentyps ausschließlich die beiden Parameter *true* oder *false* annehmen und ist direkt mit den Containertypen verbunden. Wenn es sich um eine Einstellung oder Animation handelt, so lassen sich im weiteren Verlauf sogenannte Start- und Endframes definieren. Erneut erfolgt die Sortierung über den Eintrag *name*.

[34] eigene Darstellung
[35] vgl. Abb. 12 auf Seite 36

Task-Tabelle:

TASK

id	fk_project	fk_container	name
1	2	1	design
2	2	1	modelsheet
3	2	1	modeling
4	2	1	shading
5	2	1	rigging

Abb. 17) Ausschnitt aus der Tabelle TASK[36]

Die Task-Tabelle speichert die zuvor festgelegten Aufgaben eines jeden Containers. Wenn ein neues Objekt in der Content-Tabelle angelegt wird, werden entsprechend der Task-Tabelle alle Aufgaben automatisch generiert. Wie im vorherigen Kapitel bereits erörtert, handelt es sich hierbei ausschließlich um redundante Daten. Die einhergehende enorme Zeitersparnis beim Erstellen der Jobs legitimiert allerdings den Sinn und Zweck dieser Tabelle. Die beiden Fremdschlüssel dienen wieder dem Referenzieren der entsprechenden Tabellen. In der Spalte *name* werden die Bezeichnungen der zuvor definierten Aufgaben abgelegt.

Content-Tabelle:

CONTENT

id	fk_project	fk_container	name	shot	start_frame	end_frame
1	2	1	edgar	false		
2	2	1	bruno	false		
3	2	1	luzi	false		
4	2	5	shot_001	true	0	217
5	2	5	shot_002	true	218	450

Abb. 18) Ausschnitt aus der Tabelle CONTENT[37]

Die Content-Tabelle speichert alle in den Containern befindlichen Objekte. Dies können beispielsweise Charaktere oder die Kameraeinstellungen eines Films sein. Die ersten beiden Spalten übernehmen dabei erneut die bekannten Funktionen. In der Spalte *shot* wird hinterlegt, ob das Objekt eine Timeline hat, also über Startframe und Endframe verfügt. Diese beiden Informationen werden bei Bedarf ganzzahlig in den letzten beiden gleichnamigen Spalten gesichert. Wann immer ein neuer Eintrag

[36] eigene Darstellung
[37] eigene Darstellung

angelegt wird, erstellt eine Routine im Hintergrund automatisch die entsprechenden Aufgaben in der Job-Tabelle, welche aus der Task-Tabelle referenziert werden. Angenommen, die in einem Kurzfilm auftretende Figur *edgar* (aus dem Container *characters*) beinhaltet der Einfachheit halber die vier Aufgaben *design*, *modeling*, *shading* und *rigging*.[38] So werden beim Erstellen dieses Eintrags sofort vier Datensätze in der nachfolgend erläuterten Job-Tabelle generiert. An dieser Stelle lässt sich nun auch der Vorteil der zusätzlich angelegten Task-Tabelle erkennen. Wäre diese nicht vorhanden, so müsste der Benutzer bei jedem neuen Objekt die zugehörigen Aufgaben mit angeben.

Job-Tabelle:

JOB

id	fk_project	fk_container	fk_content	fk_artist	name	status	start_date	end_date
1	2	1	1	1	design	6	04.01.2010	06.01.2010
2	2	1	1	1	modelsheet	6	04.01.2010	06.01.2010
3	2	1	1	4	modeling	2	04.01.2010	11.01.2010
4	2	1	1	4	shading	2	04.01.2010	15.01.2010
5	2	1	1	8	rigging	7	04.01.2010	28.01.2010

id	workhours	shot	start_frame	end_frame	image	comment
1	10	false			edgar/scribble.tga	
2	6	false			edagr/modelsheet.tga	Abnahme von v2
3		false				
4		false				
5		false				

Abb. 19) Ausschnitt aus der Tabelle JOB[39]

Diese Tabelle stellt in vielerlei Hinsicht den Kern der Datenbankanwendung dar. Hier werden alle Aufgaben angelegt, die daraufhin von den Mitarbeitern bearbeitet werden müssen. Auf die Analogie des Windows Explorers bezogen handelt es sich bei den Einträgen der Job-Tabelle um die Dateien des Projektsystems. Alle von den Projektleitern und Artists benötigten Informationen zu ihren Aufgaben sind in dieser Tabelle enthalten. In den ersten vier Spalten finden sich erneut die gewohnten Informationen für das Referenzieren. Über den Eintrag *name* kann der Job eindeutig identifiziert werden. Der Wert *status* speichert den aktuellen Bearbeitungsstand der Datei. Dieser Status hat eine globale Gültigkeit, entsprechend ist für jeden Benutzer der Anwendung

[38] vgl. Abb. 12 auf Seite 36
[39] eigene Darstellung

der Status derselbe. Bei dem Datenbankeintrag handelt es sich allerdings nur um einen Index, der den entsprechenden Bearbeitungsstand wiedergibt (siehe Abb. 20).

Index	Bezeichnung	Beschreibung
0	not assigned	bei Erstellung vorgegebener Status
1	not used	wird nicht verwendet
2	in progress	befindet sich in Bearbeitung
3	for approval / rendering	bereit zur Abnahme / befindet sich im Rendering
4	client approval	wartet auf Abnahme des Kunden
5	approved / rendered	wurde abgenommen / wurde gerendert
6	clean / checked	Szene wurde aufgeräumt / Rendering wurde geprüft
7	on hold	befindet sich in Wartestellung
8	ready for render	bereit zum Rendern
9	closed	ist abgeschlossen

Abb. 20) Übersicht bezüglich der Statusvergabe[40]

Für die Aufgaben, welche zur Gruppe des Arbeitsprozesses *Rendering* gehören, weichen die Statusmöglichkeiten – aufgabenbedingt – teilweise ab. Der Status *not assigned* wird automatisch beim Erstellen neuer Inhalte eingetragen und soll signalisieren, dass noch eine Bearbeitung durch den Supervisor oder Projektleiter erfolgen muss. Eine Änderung des Status auf *in progress* erfolgt in der Regel erst durch die Zuweisung des Artists.

Die Spalte *start_date* kennzeichnet das Datum, zu dem die Aufgabe erstmalig den Status *in progress* erhält, während in *end_date* gespeichert wird, wann die Fertigstellung des Jobs durch den ausführenden Mitarbeiter erwartet wird. In die Spalte *workhours* werden von diesem Mitarbeiter, nachdem die Aufgabe erledigt ist, die benötigten Arbeitsstunden eingetragen. Die Werte für die folgenden drei Spalten *shot*, *start_frame* und *end_frame* werden zunächst aus der Content-Tabelle übernommen, lassen sich aber bei Bedarf für jeden Job separat ändern. Im Feld *image* wird der Pfad zu einem möglicherweise verknüpften Bild hinterlegt. An einen Rendering-Job lässt

[40] eigene Darstellung, Inhalt wurde in einem persönlichen Gespräch mit Georg Sebastian Dressler (Supervisor bei morro images GmbH & Co. KG) gemeinsam erarbeitet, 29.01.2010

sich beispielsweise das fertig gerenderte Bild anheften, sodass die Abnahme zügig erfolgen kann. In der Spalte *comment* können vom Projektleiter oder Supervisor jobspezifische Kommentare eintragen werden.

Artist-Tabelle:

ARTIST

id	name	last_name	street	city	phone	mobile_phone	email
1	Dennis	Engler	Pfirsichweg 8	14542 Werder	0331 123456	0161 123456	de@morro.com
2	Matthias	Haase	Straße 1	12345 Stadt	0331 123456	0161 123456	mh@morro.com
3	Florian	Müller	Straße 2	12345 Stadt	0331 123456	0161 123456	fm@morro.com
4	Georg	Schulz	Straße 3	12345 Stadt	0331 123456	0161 123456	gs@morro.com
5	Christian	Mehring	Straße 4	12345 Stadt	0331 123456	0161 123456	cm@morro.com

id	position	status	expertise	day_rate	usergroup	cp	dap	eap	cu
1	Pipeline	active	programming	0 €	user	false	false	false	false
2	CEO	active	allround	0 €	projectmanager	true	true	false	false
3	Poe | modeling	active	modeling	0 €	user	false	false	false	false
4	Poe | rigging	active	modeling, rigging	0 €	supervisor	false	false	false	false
5	Admin	active	vfx, modeling	0 €	admin	false	true	true	false

id	dau	dci	su	cc	login	password
1	false	false	false	false	de_login	de_password
2	true	true	true	true	mh_login	mh_password
3	false	false	false	false	fm_login	fm_password
4	true	false	true	true	gs_login	gs_password
5	true	false	false	false	cm_login	cm_password

Abb. 21) Ausschnitt aus der Tabelle ARTIST[41]

In dieser Tabelle werden alle Informationen zu den Mitarbeitern gespeichert. In den ersten sechs Spalten werden grundlegende Daten, wie der vollständige Name und die Adresse hinterlegt. Darüber hinaus können jeweils eine Festnetz- und Handynummer eingetragen werden. Zusammen mit der Email-Adresse bilden sie die Kontaktdaten des Mitarbeiters. Die Spalte *position* beschreibt üblicherweise die Aufgabe, welche der Artists innerhalb des Unternehmens übernimmt. Über den *status* kann das Konto aktiviert bzw. deaktiviert werden. Dies ist insofern hilfreich, als dass Freelancer, die häufig für verschiedene kurzzeitige Aufträge gebucht werden, nicht jedes Mal, wenn sie das Unternehmen für eine Zeit verlassen, aus dem System gelöscht werden müssen. Unter *expertise* lassen sich die Fähigkeiten des Mitarbeiters speichern. Da diese Tabelle auch als Bibliothek für Artists verwendbar ist, lässt sich so zügig

[41] eigene Darstellung

abschätzen, ob diese Person für das nächste Projekt eingestellt werden kann. Dabei kennzeichnet *day_rate* den Tagessatz des Mitarbeiters. Die folgenden neun Spalten sind für die Rechteverwaltung von Bedeutung und werden in einem späteren Kapitel beschrieben. Die Informationen *login* und *password* sind für die Anmeldung am Programm notwendig und werden beim Autorisierungsvorgang abgeglichen.

Communicator-Tabelle:

COMMUNICATOR

id	fk_job	artist	date	text	status
1	1	Matthias Haase	04.01.2010	Beginn des Jobs	2
2	1	Dennis Engler	05.01.2010	bereit zur Abnahme	3
3	1	Matthias Haase	05.01.2010	Szene aufräumen	2
4	1	Dennis Engler	06.01.2010	alles erledigt	6
5	1	Matthias Haase	06.01.2010	Job wurde beendet	9

Abb. 22) Ausschnitt aus der Tabelle COMMUNICATOR[42]

Die Communicator-Tabelle speichert alle Einträge des gleichnamigen Moduls, das eine Art Kommentarfunktion darstellt. Unter *artist* wird der Verfasser des Kommentars hinterlegt. Die Spalte *date* sichert das Datum der Erstellung jedes Kommentars und unter *text* wird der Inhalt gespeichert. Der Wert *status* beinhaltet den Fortschritt der Aufgabe zum Zeitpunkt der Erstellung. Die Statusvergabe erfolgt nach demselben Prinzip wie vorher erläutert mit dem einen Unterschied, dass es einen weiteren Status gibt, der *comment* heißt. Dieser verändert den Status des Jobs bei Erstellung des Kommentars nicht. Ansonsten wird beim Schreiben einer neuen Nachricht automatisch der mitgegebene Status auch auf die Aufgabe übertragen. So lässt sich genau zurückverfolgen, welcher Mitarbeiter zu welcher Zeit Änderungen am Job vorgenommen hat. Dies ist insbesondere zur Kontrolle von Abnahmen sehr nützlich.

[42] eigene Darstellung

4.2 Anbindung an die Anwendung

4.2.1 FileMaker API für PHP

Wie bereits erwähnt verfügt PHP über keinerlei Möglichkeiten zur direkten Kommunikation mit FileMaker. Aus diesem Grund ist eine API nötig, die PHP um weitere Klassen erweitert, um eben jene Funktion zu gewährleisten. Diese API nennt sich *FileMaker API für PHP* und wird vom Hersteller des DBMS bereit gestellt. An dieser Stelle sei erwähnt, dass noch mindestens eine weitere Schnittstelle existiert, die sich desselben Problems annimmt. Beide unterscheiden sich hinsichtlich ihrer Syntax aber nur marginal.[43] Aufgrund der besseren Dokumentation und des im Bedarfsfall vorhandenen Supports seitens FileMaker, wurde die *FileMaker API für PHP* verwendet. Im Folgenden werden einige typische Szenarien für diese Schnittstelle anhand von Beispiel-Codes erläutert.

Initialisierung des FileMaker-Objekts:

```
1 require_once ('FileMaker.php');
2 $fm = new FileMaker("Database", "Host", "User", "Password");
```

Abb. 23) Initialisierung des FileMaker-Objekts[44]

Allen weiteren Anweisungen voran geht die Initialisierung des FileMaker-Objekts (siehe Abb. 23). Hierbei muss für *Database* die entsprechende Datenbankdatei auf dem FileMaker-Server angegeben werden. Für *Host* ist die Netzwerkadresse, auf dem diese Datei liegt, einzutragen. Über *User* und *Password* erfolgt die Anmeldung an der durch den FileMaker-Server bereitgestellten Datenbank über die Web Publishing Engine. Diese Anmeldedaten dürfen nicht mit den Konten der Mitarbeiter verwechselt werden. Nach erfolgreicher Initialisierung stehen einem alle Klassen der *FileMaker API für PHP* zur Verfügung. Andernfalls können die Befehle und Methoden nicht kompiliert werden.

[43] ein Vergleich beider Schnittstellen findet sich unter: http://jonathanstark.com/web-publishing-with-filemaker-and-php.php, 20.05.2010

[44] eigene Darstellung, vgl. Blassl ; Früchtel: "Das neue FileMaker Forum: Einsatz von Web 2.0, FM PHP API und XML", http://downloads.filemaker.de/konferenz2006/FileMakerForum_Fruechtel_Blassl.pdf, S. 13, 20.05.2010

Auslesen von Datensätzen:

```
1 $comFind = $fm -> newFindCommand("Table");
2 $comFind -> addFindCriterion("Criterion" , "=" . $variable);
3 $result = $comFind -> execute();
```

Abb. 24) Anweisungen für das Auslesen von Datensätzen[45]

Nun können unter anderem auch ein oder mehrere Datensätze ausgelesen werden (siehe Abb. 24). Als erstes muss dazu ein neues Suchkommando initialisiert werden, wobei *Table* für die zu durchsuchende Tabelle steht. Danach wird ein Suchkriterium definiert, entsprechend ist *Criterion* durch eine Spalte der Tabelle zu ersetzen. Dahinter findet sich der Vergleichsoperator – in diesem Beispiel wird nach allen Datensätzen gesucht, die identisch mit dem Kriterium sind. Dem Operator folgt nun das Suchwort, wobei es sich in der Regel um eine Variable handelt. Es könnte aber auch eine Konstante eingesetzt werden. Mit Hilfe der letzten Zeile wird der Suchprozess gestartet und das Ergebnis in der Variable *$result* gespeichert. Der Inhalt dieser Variablen muss im Anschluss daran ausgewertet werden.

Erstellen neuer Datensätze:

```
1 $values = array('name' => "Dennis", 'last_name' => "Engler");
2 $rec =& $fm -> createRecord('Table', $values);
3 $result = $rec -> commit();
```

Abb. 25) Anweisungen für das Erstellen neuer Datensätze[46]

Für das Anlegen eines neuen Datensatzes in der FileMaker-Datenbank gibt es zwei Möglichkeiten. Für die erste Variante wird zunächst ein Array mit den zu füllenden Feldern und deren Werten angelegt (siehe Zeile 1 in Abb. 25). Anschließend wird ein neuer *Record* erzeugt, dem das zuvor erstellte Array als Parameter übergeben wird. Die zweite Variante initiiert zuerst den *Record*, welchem nur die Tabelle mitgegeben wird. Erst danach werden mittels *$rec->setField('name', "Dennis")* die Werte zugewiesen. In beiden Fällen wird abschließend der Erstellungsprozess, wie in der letzten Zeile dargestellt, gestartet.

[45] eigene Darstellung, vgl. Blassl ; Früchtel: "Das neue FileMaker Forum: Einsatz von Web 2.0, FM PHP API und XML", http://downloads.filemaker.de/konferenz2006/FileMakerForum_Fruechtel_Blassl.pdf, S. 14f, 20.05.2010

[46] eigene Darstellung, vgl. Blassl ; Früchtel: "Das neue FileMaker Forum: Einsatz von Web 2.0, FM PHP API und XML", http://downloads.filemaker.de/konferenz2006/FileMakerForum_Fruechtel_Blassl.pdf, S. 16, 20.05.2010

Ändern vorhandener Datensätze:

```
1 $editCom = $fm -> newEditCommand("Table", "DatabaseID");
2 $editCom -> setField("Column", $variable);
3 $result = $editCom -> execute();
```

Abb. 26) Anweisungen für das Ändern vorhandener Datensätze[47]

Um einen bereits vorhanden Datensatz abzuändern wird zunächst das Edit-Objekt initialisiert. Diesem Objekt muss neben *Table* eine weitere Variable übergeben werden, die *DatabaseID*. Dies ist eine unveränderbare, von FileMaker für jeden Datensatz automatisch generierte Indexvariable. Sie kann mit Hilfe des Kommandos *$record->getRecordId()* ausgelesen werden und ermöglicht die eindeutige Bestimmung eines Datensatzes. Danach folgt eine Auflistung von Anweisungen zum Ändern der gewünschten Werte. Diese Funktionen benötigen als Übergabeparameter die Spalte der Tabelle und den neuen Wert. Im letzten Schritt wird der Befehl ausgeführt und das Ergebnis der Anfrage in einer Variablen gespeichert. Diese Variable kann dann zur späteren Fehlerauswertung herangezogen werden.

[47] eigene Darstellung, vgl. Blassl ; Früchtel: "Das neue FileMaker Forum: Einsatz von Web 2.0, FM PHP API und XML", http://downloads.filemaker.de/konferenz2006/FileMakerForum_Fruechtel_Blassl.pdf, S. 17f, 20.05.2010

4.2.2 PHP-Klassen

Für die Anbindung an die Datenbank werden insgesamt sechs PHP-Dateien verwendet (siehe Abb. 27). Dies geschah allein aus Gründen der Übersicht – theoretisch hätte der Programmcode auch in einer einzigen Klasse mit entsprechend mehr Methoden geschrieben werden können.

Klasse	Beschreibung
dbAccess.php	die integrierten Methoden steuern das Auslesen der Datensätze
dbCreate.php	wird für die Erstellung neuer Datensätze verwendet
dbDelete.php	beinhaltet Methoden für das Löschen von Datensätzen
dbEdit.php	steuert das Ändern von vorhandenen Datensätzen
dbUpload.php	regelt den Upload von Media-Dateien
dbView.php	enhält Methoden für die Bereitstellung der detaillierten Tabellenansicht

Abb. 27) Übersicht über die verwendeten PHP-Klassen[48]

Die *dbAccess.php* wird für fast alle Anfragen verwendet, die mit dem Auslesen von Datensätzen verbunden sind. Anhand eines Beispielcodes zu dieser Klasse wird die Vorgehensweise bei der Programmierung aller weiteren Klassen erläutert.

Wie bereits im vorherigen Kapitel erklärt, wird zunächst ein FileMaker-Objekt erzeugt und der Zugriff auf die Datenbank realisiert (siehe Abb. 28 auf der nachfolgenden Seite). Dem folgt die Generierung einer Variablen, die als Wert aus der ActionScript-Anwendung ein sogenanntes *Layout* erhält. Über diese Variable wird im weiteren Programmablauf mit Hilfe einer *switch case* Anweisung jene Methode ermittelt, die für die entsprechende Datenbankabfrage benötigt wird. Im vorliegenden Ausschnitt sollen die Informationen zum Generieren der Tabellenzeile(n) aus der Hierarchiestufe *content* aus der Datenbank ausgelesen werden. Zudem wird eine weitere Variable *theend* deklariert und mit dem vorgegebenen Wert *0* versehen. Diese Variable wird für die spätere Erzeugung des Ausgabestroms verwendet. Innerhalb der *case*-Struktur wird abermals eine Variable *fk_container* erzeugt, deren Wert aus dem vom Frontend

[48] eigene Darstellung

stammenden Datenstrom erfasst wird. Diese Variable wird in der folgenden Such-Anweisung als Suchkriterium verwendet. Darauf folgt eine Anweisung zum Sortieren der erhaltenen Datensätze nach einem spezifischen Kriterium (in diesem Fall wird nach der Spalte *name* sortiert).

```php
<?php

   require_once ('FileMaker.php');

   $fm = new FileMaker();
   $fm -> setProperty('database', 'postproductionPipeline_1_0');
   $fm -> setProperty('hostspec', 'http://localhost');
   $fm -> setProperty('username', 'user');
   $fm -> setProperty('password', 'password');

   $layout = $_GET["layout"];

   $theend = 0;

   switch ($layout) {

      case "content":
         $fk_container = $_POST["fk"];

         $contentFind = $fm -> newFindCommand("content");
         $contentFind -> addFindCriterion("fk_container", "=" . $fk_container);
         $contentFind -> addSortRule('name', 1, FILEMAKER_SORT_ASCEND);
         $contentResult = $contentFind -> execute();

         if (FileMaker::isError($contentResult)) {
            echo "Error$theend=" . $contentResult -> code . "&";
            $theend++;
            echo "theend=$theend";
         } else {

            $contentRecords = $contentResult -> getRecords();
            $contentRecord = $contentRecords[0];

            foreach($contentRecords as $contentRecord) {
               echo "ID$theend=" . $contentRecord -> getField('id') . "&";
               echo "DatabaseID$theend=" . $contentRecord -> getRecordId() . "&";
               echo "Name$theend=" . $contentRecord -> getField('name') . "&";
               $theend++;
            }
            echo "theend=$theend";
         }
      break;

      //weitere Abfragen

   }
?>
```

Abb. 28) Auszug aus der dbAccess.php[49]

Anschließend wird das Such-Kommando ausgeführt und die Datenbank nimmt eine entsprechende Suche nach den übermittelten Daten vor. Die nachfolgenden Befehle wurden implementiert, um im Falle eines Fehlers bei der Datenbankabfrage den

[49] eigene Darstellung

Errorcode an ActionScript zu übergeben (Zeile 25 - 29 in Abb. 28). Mittels dieses Fehlercodes können dann spezifische Verhaltensweisen implementiert werden.[50] Wenn keine Fehlermeldung von FileMaker zurückkommt, wird das Ergebnis der Abfrage in ein Array geschrieben. Schlussendlich wird ein Ausgabestrom generiert, der nun durch die ActionScript-Applikation ausgelesen und weiterverarbeitet werden kann.

Als Ergebnis dieser Abfrage sind, wie bereits angedeutet, zwei grundlegend verschiedene Varianten zu erwarten. Gesetzt dem Fall, der gesuchte Datensatz konnte gefunden werden, so wird ein Ausgabestrom erzeugt, der sich am ehesten mit einem mehrdimensionalen Array vergleichen lässt (siehe Abb. 29). Dieser Ausgabestrom lässt sich in AS3 mittels eines sogenannten *http-Requests* einlesen. Wenn die Datenbankabfrage einen Fehler erzeugt wird ein Ausgabestrom erzeugt, der den entsprechenden Errorcode von FileMaker enthält.

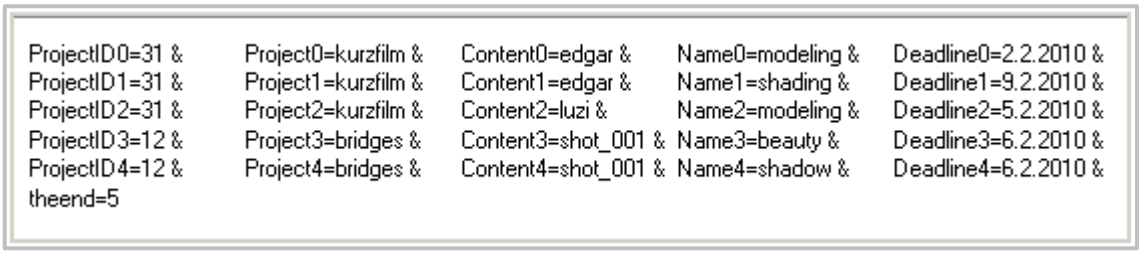


Abb. 29) Ausgabestrom einer Datenbankabfrage zum JobView-Modul[51]

[50] eine Auflistung aller von FileMaker verwendeten Errorcodes findet sich unter: "FileMaker 11: Custom Web Publishing mit PHP", http://filemaker.de/support/product/docs/fms/fms11_cwp_php_de.pdf, S. 57ff, 19.05.2010

[51] eigene Darstellung, aus Gründen der Übersicht erfolgt die Darstellung formatiert

4.3 Frontend Programmierung

4.3.1 Allgemeines zur Implementierung mit ActionScript

Wie im Kapitel *Technologische Eckpfeiler* bereits erwähnt, wird ActionScript 3.0 in Verbindung mit der Flash-Entwicklungsumgebung zur Programmierung des Frontend verwendet. Im Folgenden wird auf die grundlegenden Methoden zur Implementierung mit ActionScript 3.0 eingegangen.

Variablen sind absolut unentbehrlich für objektorientierte Programmiersprachen. Auch ActionScript macht hier keine Ausnahme. In AS3 können Variablen mit weiteren Attributen, wie beispielsweise *public* oder *private* versehen werden (siehe Abb. 30). Mittels solcher Attribute kann gesteuert werden, ob die Variable auch in anderen Klassen oder Methoden, als der in der sie erstellt wurde, verwendet werden kann. Darüber hinaus ist es möglich die Variablendeklaration innerhalb einer Zeile durchzuführen, indem der gewünschte Wert gleich mitgegeben wird (siehe Zeile 4 in Abb. 30). Seit Version 3.0 können in ActionScript auch Konstanten deklariert werden. Dies geschieht analog zu den Variablen. Darüber hinaus muss der Variablen oder Konstanten ein Datentyp zugewiesen werden. Dieser Typ definiert, welche Operationen auf und mit der Variablen sinnvoll und zulässig sind.[52]

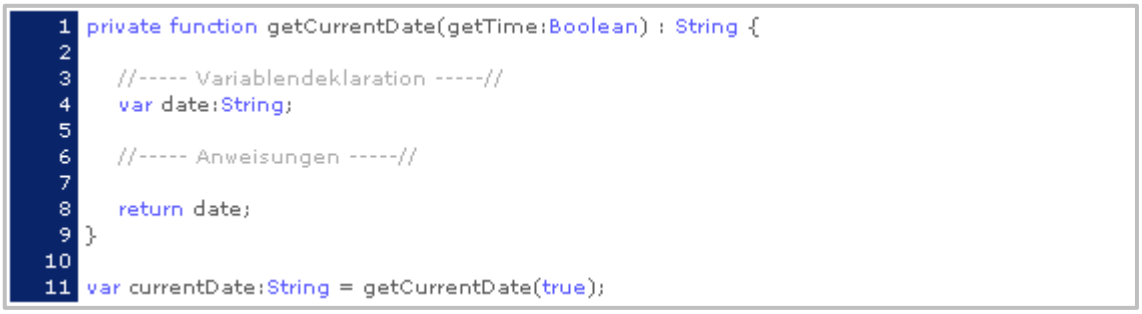
```
1 public var number:int;
2 number = 25;
3
4 private var word:String = "Hallo Welt";
5
6 const blubber:Boolean = true;
```

Abb. 30) Verwendung von Variablen und Konstanten in ActionScript 3.0[53]

Methoden werden in AS3 verwendet, um häufig auftretende Funktionen zu definieren. Es wird zwischen Methoden mit und ohne Rückgabewert unterschieden. Im Beispiel findet sich eine Funktion, die das aktuelle Datum mittels Rückgabewert ausgeben soll (siehe Abb. 31 auf der nachfolgenden Seite). Genau wie Variablen verfügen auch Methoden über ein Attribut, um zu bestimmen, ob die Methode auch außerhalb ihrer

[52] vgl. Braunstein, R. ; Wright, M. H. ; Noble, J. J.: "ActionScript™ 3.0 Bible", S. 7ff, Indianapolis 2007
[53] eigene Darstellung

Klasse verwendet werden kann. Nach dem Methodennamen folgen eventuell benötigte Übergabeparameter und schließlich der Datentyp. Im Fall einer Methode ohne Rückgabewert würde stattdessen der Ausdruck *void* Verwendung finden. Üblicherweise findet die Variablendeklaration zu Beginn der Methode statt, dies muss jedoch nicht zwangsläufig so sein. Danach folgen dann die entsprechenden Anweisungen, die das gewünschte Ergebnis erzeugen. Wenn es sich um eine Methode mit Rückgabewert handelt, wird mittels *return* die Ausführung der Methode beendet und über eine nachgestellte Variable der entsprechende Wert zurückgegeben. Der Aufruf einer Methode erfolgt über ihren Namen, dem bei Bedarf in Klammern Übergabeparameter folgen (siehe Zeile 11 in Abb. 31).



```
1  private function getCurrentDate(getTime:Boolean) : String {
2
3     //----- Variablendeklaration -----//
4     var date:String;
5
6     //----- Anweisungen -----//
7
8     return date;
9  }
10
11 var currentDate:String = getCurrentDate(true);
```

Abb. 31) Verwendung von Methoden in ActionScript 3.0[54]

Wie in objektorientierten Programmiersprachen üblich werden Klassen als abstraktes Modell eingesetzt, um in der Regel wiederkehrende Objekte zu definieren. In einem typischen Klassenmodell wird zunächst der Speicherort in Relation zum Flash-Dokument benötigt (siehe Zeile 1 in Abb. 32 auf Seite 54). Danach können weitere Klassen eingebunden werden. Eine solche Importfunktion könnte beispielsweise folgendermaßen ausschauen: *import components.quickMenu.BasicWindow*. Die ersten beiden Informationen *components* und *quickMenu* repräsentieren den Speicherort, respektive das zuvor definierte *package*. Es müssen jedoch nicht zwingenderweise andere Klassen importiert werden. Danach folgt die Klassendefinition. In der Regel wird vor dem Klassennamen ein Attribut verwendet, welches kongruent zur Deklaration der Variablen die Zugriffrechte steuert. Darüber hinaus können Klassen auch Methoden anderer Klassen erben. Dies geschieht mittels *extends*, worauf der Name der Klasse folgt, von der geerbt werden soll. Diese Klasse muss dann auch

[54] eigene Darstellung

zwingend vorher importiert werden. Die Initialisierungsmethode (siehe Zeile 9 - 13 in Abb. 32) wird beim Aufruf der Klasse automatisch durchlaufen. Weitere Methoden können danach definiert werden, müssen allerdings entweder durch die Klasse selbst oder innerhalb anderer Klassen der Anwendung, manuell aufgerufen werden.

```
1  package packageName {
2
3     //----- import anderer Klassen -----//
4
5     public class className extends MovieClip {
6
7        //----- Variablendeklaration -----//
8
9        public function className() {
10
11          //----- Hauptmethode der Klasse -----//
12
13       }
14
15       //----- weitere private oder public Methoden
16
17 }
```

Abb. 32) typischer Klassenaufbau mit ActionScript 3.0[55]

4.3.2 Funktionen der Postproduction-Pipeline

Zu den wichtigsten Funktionen innerhalb des Frontends der Postproduction-Pipeline gehört die sogenannte *customWindow* Klasse. Sie wird für das Einblenden aller Dialogfenster verwendet. Diese Fenster kommen jedes Mal dann zum Einsatz, wenn der Benutzer ein neues Objekt erstellen möchte. Dieser Vorgang erfolgt immer über dieselbe Schaltfläche und ist abhängig davon, welche Auswahl im OverView-Modul angezeigt wird. Befindet sich der User beispielsweise in der Mitarbeiterliste und hat das Recht neue Benutzer hinzuzufügen, dann erscheint beim Klick auf den Button das Dialogfenster für das Anlegen eines neuen Mitarbeitereintrags (siehe Abb. 33 auf der nachfolgenden Seite). Zur Abfrage der benötigten Daten werden verschiedene Eingabemodule eingeblendet. Über die Schaltfläche *ok* wird der Eintrag in der Datenbank gespeichert, andernfalls gehen die Eintragungen verloren und das Fenster schließt sich.

[55] eigene Darstellung

Abb. 33) Erstellung eines neuen Benutzers mittels eines Dialogfensters[56]

Umgesetzt wurde diese Funktion folgendermaßen: In der Hauptklasse der Anwendung (die sogenannte *main.as*) wurde eine Methode *createItems* implementiert, mittels derer die Initialisierung der *customWindow* Klasse erfolgt (siehe Abb. 34 auf Seite 56). Zunächst wird ein neues Objekt erstellt, wobei als Parameter die Pfade der PHP-Skripte mit übergeben werden. Anschließend folgt eine *switch case* Struktur, welche den aktuellen Inhalt des OverView-Moduls *currentState* abfragt. Anhand dieser Information werden unterschiedliche Methoden aus der *customWindow* Klasse ausgeführt. Teilweise ist das Übertragen weiterer Parameter erforderlich. Dies können beispielsweise verschiedene FKs sein, die beim Erstellen eines Datensatzes in der Datenbank die Beziehungen zu anderen Einträgen herstellen. Danach wird dem Window-Objekt ein EventListener hinzugefügt. Dieser EventListener wird benötigt, um in einer weiteren Methode erfassen zu können, ob die Eingaben im Dialogfenster bestätigt wurden oder ob ein Abbruch durch den User erfolgte. Zu diesem Zweck wurde eine eigene Klasse *customEvent* erstellt. Darauf folgen Anweisungen zum mittigen Platzieren des Window-Objekts innerhalb der sogenannten Stage.[57] Abschließend

[56] eigene Darstellung, kompletter Ausschnitt aus der Applikation
[57] auch Bühne genannt, definiert den Arbeitsbereich eines Flash-Dokuments

erfolgt das Verwischen aller dahinter befindlichen Elemente. Diese Funktion wurde wiederum über eine separate Klasse implementiert. Zusätzlich zum Verwisch-Effekt wird mittels einer Colormatrix der Hintergrund entsättigt. Beim Schließen des Fensters wird der EventListener vom Window-Objekt und selbiges von der Stage entfernt. Weiterhin wird die Manipulation am Hintergrund wieder rückgängig gemacht.

```
3967 private function createItems(e:MouseEvent) : void {
3968
3969    //----- neue Window-Klasse -----//
3970    newWindow = new customWindow(phpCreate, phpEdit);
3971
3972    switch (currentState) {
3973
3974       case 1:
3975          newWindow.newProject(user, userID, managerArray, managerIDArray);
3976       break;
3977
3978       case 2:
3979          newWindow.newContainer(supervisorArray, supervisorIDArray, projectFK);
3980       break;
3981
3982       case 3:
3983          newWindow.newContent(projectFK, containFK, containShot);
3984       break;
3985
3986       case 4:
3987          newWindow.newTask(projectFK, containFK);
3988       break;
3989
3990       case 5:
3991          newWindow.newArtist();
3992       break;
3993    }
3994
3995    //----- EventListener hinzufügen und Window platzieren -----//
3996    newWindow.addEventListener(customEvent.CLICK_TARGET, newWindowHandler);
3997    newWindow.x = stage.stageWidth / 2 - newWindow.width / 2;
3998    newWindow.y = stage.stageHeight / 2 - newWindow.height / 2;
3999    stage.addChild(newWindow);
4000
4001    //----- Hintergrund blurren -----//
4002    blurBackground = new Blurring(morroPipe_mc, stage.height, stage.width);
4003    var colorMatrixGrey:ColorMatrixFilter = new ColorMatrixFilter(matrixGrey);
4004    morroPipe_mc.filters = [colorMatrixGrey];
4005 }
```

Abb. 34) Methode zum Initialisieren der Window-Klasse[58]

Diese dargelegte Vorgehensweise zieht sich durch die gesamte Programmstruktur. Die Verwendung möglichst vieler Klassen dient einerseits der Übersichtlichkeit bei der Handhabung umfassender Programmcodes. Andererseits ermöglicht sie wie bereits erwähnt auch das Wiederverwenden bereits implementierter Funktionen.

[58] eigene Darstellung, Ausschnitt aus der main.as

Ein weiteres wichtiges Element, stellen die interaktiven Schaltflächen dar. Auch wenn man vermuten würde, dass es sich hierbei um ein überflüssiges Gimmick handelt, so erfüllen sie doch einen wichtigen Zweck. Sie helfen dem User dabei zu registrieren, welche Informationen sich mit der Maus anklicken lassen. Besonders in einer solch komplexen Applikation, in der viele Informationen, wie beispielsweise die Zeilen einer Übersichtstabelle, als Schaltfläche für weitere Funktionen dienen, ist eine solche Vorgehensweise von Vorteil.

In der Flash-Entwicklungsumgebung wird dazu zunächst ein neues Dokument erstellt, in dem alle benötigten Grafiken an einem beliebigen Platz innerhalb der Arbeitsfläche platziert werden (siehe Abb. 35). Am Beispiel der *customRow* Klasse lässt sich erkennen, dass nicht nur die Grafiken an sich, sondern auch Elemente wie eine sogenannte *Hitbox*, mit der die entsprechenden EventListener später verknüpft werden, erforderlich sind. Textfelder werden für das Anzeigen von dynamischen Informationen verwendet.

Abb. 35) Grafiken zur Erstellung der *customRow* Klasse[59]

Anschließend wird eine entsprechende AS3 Klasse implementiert, in der alle Funktionen bezüglich der Schaltfläche beschrieben werden. Der Initialisierungs-methode werden insgesamt drei Parameter übergeben. Diese bestimmen zum einen die Breite der dargestellten Grafik und ob die Schaltflächen für das Löschen und Ändern von Einträgen vorhanden sind. Falls es sich um eine Tabellenzeile aus der Ebene *project* handelt, wird außerdem der Name des Kunden übergeben. Danach folgt, noch innerhalb der Initialisierungsmethode, das Platzieren der Grafiken und die Verknüpfung der Hitbox mit den EventListenern. In diesem Beispiel ist die Methode durch eine *if else* Anweisung zweigeteilt, weil das Platzieren der Objekte je nach

[59] eigene Darstellung

Zugriffsrecht des Benutzers unterschiedlich ausfällt. Danach folgt eine Methode mittels derer der Text geändert werden kann. Diese Methode wird von der Hauptklasse aus ausgeführt, weshalb sie als Attribut *public* verwendet. Schlussendlich folgen alle Event-Methoden, die je nach Aktion des Users ausgeführt werden (siehe Zeile 317 - 321 in Abb. 36). Sie steuern das Verhalten der Grafiken, wenn der Benutzer beispielsweise mit der Maus über den Delete-Button der Tabellenzeile fährt. So erhält der User ein optisches Feedback auf seine Aktionen.

```
  1 package components {
    .
    .   //----- import der ActionScript Klassen -----//
    .
  8   public class customRow extends MovieClip {
    .
    .     //----- Deklaration der Variablen -----//
    .
 35     public function customRow(  getWidth:uint,
 36                                 getFullAccess:Boolean = false,
 37                                 getCustomer:String = "") {
 38
 39       //----- Wenn der User Manager-Rechte besitzt -----//
 40       if (getFullAccess == true) {
    .
    .         //----- Platzierung der Grafiken -----//
    .
118         //----- EventListener für den Delete-Button -----//
119         rowDeleteHit.addEventListener(MouseEvent.MOUSE_OVER, delete_mouseOver);
120         rowDeleteHit.addEventListener(MouseEvent.MOUSE_OUT, delete_mouseOut);
121         rowDeleteHit.addEventListener(MouseEvent.MOUSE_DOWN, delete_mouseDown);
    .
    .         //----- weitere EventListener für die restlichen Schaltflächen -----//
    .
135       } else {
    .
    .         //----- Platzierung der Grafiken -----//
    .
212       }
213     }
214
215     //----- Methode zum Ändern des Textes der Tabellenzeile -----//
216     public function setLabel(getLabel:String) : void {
217       label_txt.text = getLabel;
218     }
    .
    .     //----- weitere Methoden der Row-Klasse -----//
    .
317     //----- MouseOver-Methode des Delete-Buttons -----//
318     private function delete_mouseOver(e:MouseEvent) : void {
319       rowDeleteOut.visible = false;
320       rowDeleteOver.visible = true;
321     }
    .
    .     //----- alle weiteren EventListener-Methoden -----//
    .
691   }
692 }
```

Abb. 36) Auszüge aus der *customRow* Klasse[60]

[60] eigene Darstellung

4.4 System zur Rechteverwaltung

Für das Rightsmanagement (RM) kommen nicht die traditionellen Dateizugriffsrechte eines Unix-Systems zum Einsatz. Stattdessen wurde ein eigenes System konzipiert, das auf die Bedürfnisse eines Medienproduktionsunternehmens zugeschnitten ist. Dieser Schritt ist notwendig, weil sich im Laufe einer Produktion die Zusammensetzung des Teams mehrfach ändern kann. Das verwendete RM-System wurde daher in Hinblick auf die Vererbbarkeit von Rechten entwickelt. Die verwendeten Benutzergruppen besitzen zwar vorgegebene Rechte, jedoch lassen sich diese bei Bedarf ändern. Zudem sind spezielle Rechte von Nöten, die so nur in einem Medienunternehmen zum Einsatz kommen. Insgesamt wurden fünf Benutzergruppen definiert, die einen Großteil des typischen Verwendungszwecks abdecken (siehe Abb. 37).

Kürzel	Bezeichnung	User	Supervisor	Personalmanager	Projectmanager	Administrator
cp	create project					
dap	display all projects					
eap	edit all projects					
cu	create user					
dau	display all users					
dci	display contract info					
su	supervisor					
cc	create comment					

Abb. 37) Benutzergruppen mit ihren vordefinierten Rechten[61]

Jedes Recht gewährt dem User Zugriff auf spezielle Funktionen innerhalb der Anwendung. Nach erfolgreicher Autorisierung werden die Einträge bezüglich des RM aus der Datenbank in ein Array geschrieben. An gegebener Stelle innerhalb der Applikation erfolgt dann eine Abfrage dieser Werte. Nachfolgend findet eine Erläuterung der integrierten Rechte statt.

[61] eigene Darstellung

create project: Dieses Recht ermöglicht dem Besitzer das Erstellen neuer Projekte. Zugleich kann er in bereits vorhandenen Projekten als Manager zugewiesen werden. Innerhalb seiner Aufträge hat der Projektleiter vollen Zugriff auf alle Funktionen der Anwendung. Sie erlauben es ihm Aufgaben zu löschen, zu editieren und neue Jobs anzulegen. Darüber hinaus ist er dafür verantwortlich, den Containern jeweils einen Supervisor zuzuweisen, falls er diese Funktion nicht selbst übernimmt.

display all projects: Es erlaubt dem Benutzer in alle laufenden Projekte im Unternehmen hineinzuschauen. Alle User ohne dieses Recht können ausschließlich jene Projekte einsehen, mit denen sie in direktem Zusammenhang stehen – sei es als Artist oder als Supervisor. Dieses Recht wurde einerseits integriert, um den Großteil der Mitarbeiter nicht mit übermäßig vielen Informationen zu konfrontieren. Andererseits ist es auch nicht erforderlich, dass projektfremde Mitarbeiter Einblick in jeden Auftrag erhalten. Gerade Administratoren oder CEOs sollte dennoch die Möglichkeit eingeräumt werden alle Projekte betrachten zu können.

edit all projects: Es gestattet dem Besitzer projektübergreifend alle Einträge zu bearbeiten. Dieses Recht findet in der Praxis kaum Verwendung, doch sollte die grundsätzliche Möglichkeit dafür vorhanden sein. Derzeit verfügen ausschließlich der Administrator und die CEOs des Unternehmens über dieses Recht. Auf diese Weise ist es ihnen möglich auf laufende Prozesse einwirken zu können, ohne zwingend an ihnen beteiligt zu sein.

create user: Dieses Recht ermöglicht die Erstellung neuer Benutzer. Diese Aufgabe übernehmen in der Regel die Personalmanager. Beim Erstellen eines neuen Benutzers wird zunächst die Benutzergruppe *user* ausgewählt, der neue Account verfügt damit nur über die absolut grundlegendsten Rechte. Dazu gehört das Einsehen der Projekte an denen der Benutzer mitwirkt und das hoch- sowie herunterladen von Dateien innerhalb seiner Aufgaben. Natürlich lassen sich sowohl die Usergroup als auch die einzelnen Rechte sowohl beim Erstellungsprozess als auch nachträglich anpassen.

display all users: Das Benutzerkonto, welches über dieses Recht verfügt, erhält Einsicht in die Mitarbeiterliste des Unternehmens. Personalmanager können so einsehen wer für welches Projekt gebucht ist und wie seine Präferenzen aussehen. Zusätzlich erhalten auch Projektleiter und Supervisor dieses Recht, da sie dafür zuständig sind, den Mitarbeitern ihre Aufgaben zuzuweisen. Alle anderen User können ausschließlich über bereitgestellte Verknüpfungen innerhalb ihres Projekts, auf die Kontaktdaten ihrer Vorgesetzten zugreifen.

display contract info: Dieses Recht gewährt dem User Zugriff auf vertrauliche Information hinsichtlich der typischen Tagessätze der Mitarbeiter. Diese helfen bei der Kalkulation der Projekte und sollten ausschließlich von den Personalmanagern und CEOs eingesehen werden dürfen.

supervisor: Es ermöglicht allen Benutzern, die über dieses Recht verfügen, als Supervisor in den Containern eines Projektes eingetragen zu werden. Innerhalb seines Aufgabenbereichs besitzt ein Supervisor dieselben Rechte wie ein Projektleiter. Er kann also sowohl neue Aufgaben erstellen und bearbeiten, als auch das Löschen eben jener vornehmen.

create comment: Es erlaubt dem Besitzer Kommentare zu den einzelnen Aufgaben zu erstellen. In der Regel besitzen Projektleiter und Supervisor dieses Recht automatisch. Es kann allerdings stellenweise erforderlich sein, auch anderen Mitarbeitern diese Funktion zu gewähren. Deswegen wurde es als separates Recht integriert und ist nicht zwangsläufig an die beiden Manager-Rechte gekoppelt. Dabei ist das Recht auf jene Projekte beschränkt, an denen der Benutzer beteiligt ist. Eine Ausnahme kommt zum Tragen wenn der User über das *edit all projects* Recht verfügt. In diesem Fall ist er in der Lage, zu allen Aufgaben – also projektübergreifend – Kommentare zu erstellen.

4.5 Gestaltung und Layout

4.5.1 Design

Die farbliche Gestaltung einer Anwendung sollte grundsätzlich auf das Corporate Design des Unternehmens abgestimmt sein. So kann seitens der Mitarbeiter eine Bindung zwischen Programm und Unternehmen aufgebaut werden. Aufgrund der lediglich zwei Farben umfassenden Palette im untersuchten Unternehmen (schwarz und weiß), ließ sich diese Vorgabe aber nur schwer umsetzen.

Anthrazit wurde schließlich als Basis gewählt, da es am ehesten mit dem Corporate Design vereinbar war. Diese Farbe vermittelt im Allgemeinen zugleich eine gewisse Wertigkeit und soll den modernen Charakter der Anwendung unterstreichen. Um Kontraste setzen zu können, müssen darüber hinaus weitere Abstufungen gefunden werden. Daher wurde eine Auswahl verschiedener Graustufen erstellt, welche als Fundament dienen. Die Wahl fiel dabei auf besonders unaufdringliche und gefühls-neutrale Töne. Dies ist bei der Entwicklung einer Anwendung, mit der viele Benutzer tagtäglich arbeiten, von großer Wichtigkeit. Im Gegensatz zu Medien aus der Werbung geht es nicht darum Aufmerksamkeit zu erregen. Um die Lesbarkeit von Texten zu optimieren, wurde die Farbpalette um einen reinen Weiß- und Schwarzton erweitert. Da die Mitarbeiter in einem Medienproduktions-Unternehmen nicht selten die gesamte Arbeitszeit vor einem Monitor verbringen, ist es wichtig jegliche Anstrengung beim Lesen von Texten zu vermeiden und die Augen soweit wie möglich zu entlasten. Für das Setzen von Akzenten wurde ein warmer Orangeton gewählt. Andernfalls hätte die ausschließliche Verwendung von Grautönen eine sehr triste Darstellung zur Folge gehabt (siehe Abb. 38 auf der nachfolgenden Seite).

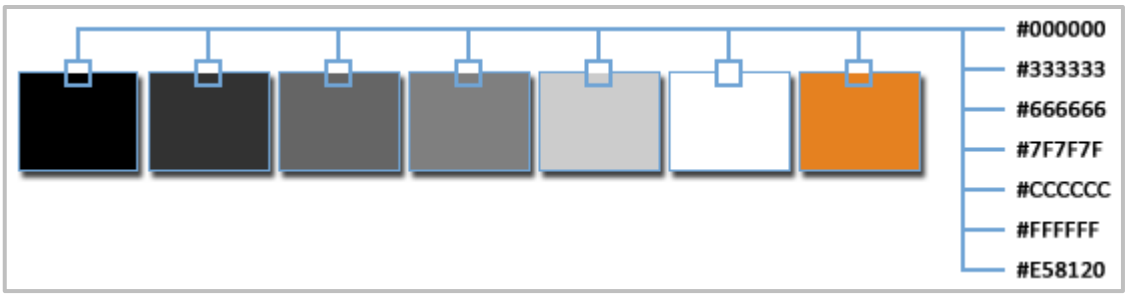


Abb. 38) Farbwerttabelle der Postproduction-Pipeline[62]

Die Auswahl der Farben zu den Statusfeldern geschah nach größtmöglicher Unterscheidbarkeit. So soll auf den ersten Blick klar zu erkennen sein, um welchen Status es sich handelt. Auch eine falsche Kalibrierung vieler Monitore erfordert es, geringe Farbabweichungen zu vermeiden, da diese andernfalls nicht wahrgenommen werden können oder schlichtweg nicht zu unterscheiden sind. Weiterhin wurde versucht jedem Status eine dem Sinn entsprechende Farbe zu geben. So sind Rottöne, ihrer allgemeinen Empfindung entsprechend, für Status verwendet worden, die Handlungsbedarf erfordern. Wohingegen grünliche Farbtöne dem Empfinden nach für abgeschlossene Prozesse stehen und in zwei Variationen für ebendiese Status verwendet wurden. Blaue Farbnuancen wirken auf die Menschen eher kalt und werden für Handlungsabschnitte genutzt, denen kein akuter Handlungsbedarf nachgeht. Die beiden Status *not assigned* und *not used* erhielten aufgrund ihrer Ausnahmestellung zwei Grautöne (siehe Abb. 39).

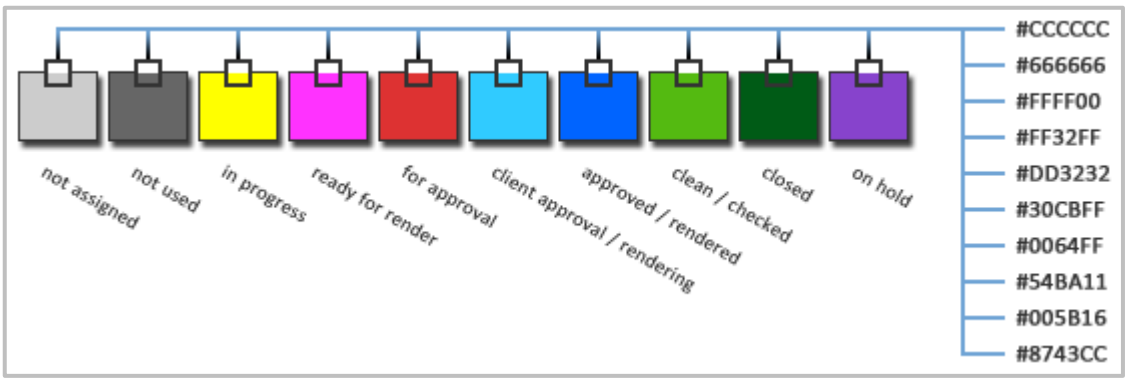


Abb. 39) Farbtabelle für die Statusfelder[63]

[62] eigene Darstellung
[63] eigene Darstellung

Der Wahl des Schrifttyps muss angemessene Beachtung geschenkt werden. Jede Software verwendet zudem andere Algorithmen zur Darstellung der vielen verschiedenen Schriftarten. Nach diversen Tests und persönlichem Empfinden, ist der Autor zu dem Entschluss gekommen, dass sogenannte *TrueType* Schriftarten unter Adobe Flash nur in unzureichender Qualität dargestellt werden. Aus diesem Grund wurde eine Systemschrift gewählt, im speziellen die Schriftart *Sans* von Microsoft. Diese garantiert eine optimale Lesbarkeit, gerade auch in Hinblick darauf, dass die Schrift innerhalb der Applikation teilweise schräg dargestellt wird.

4.5.2 Graphical User Interface

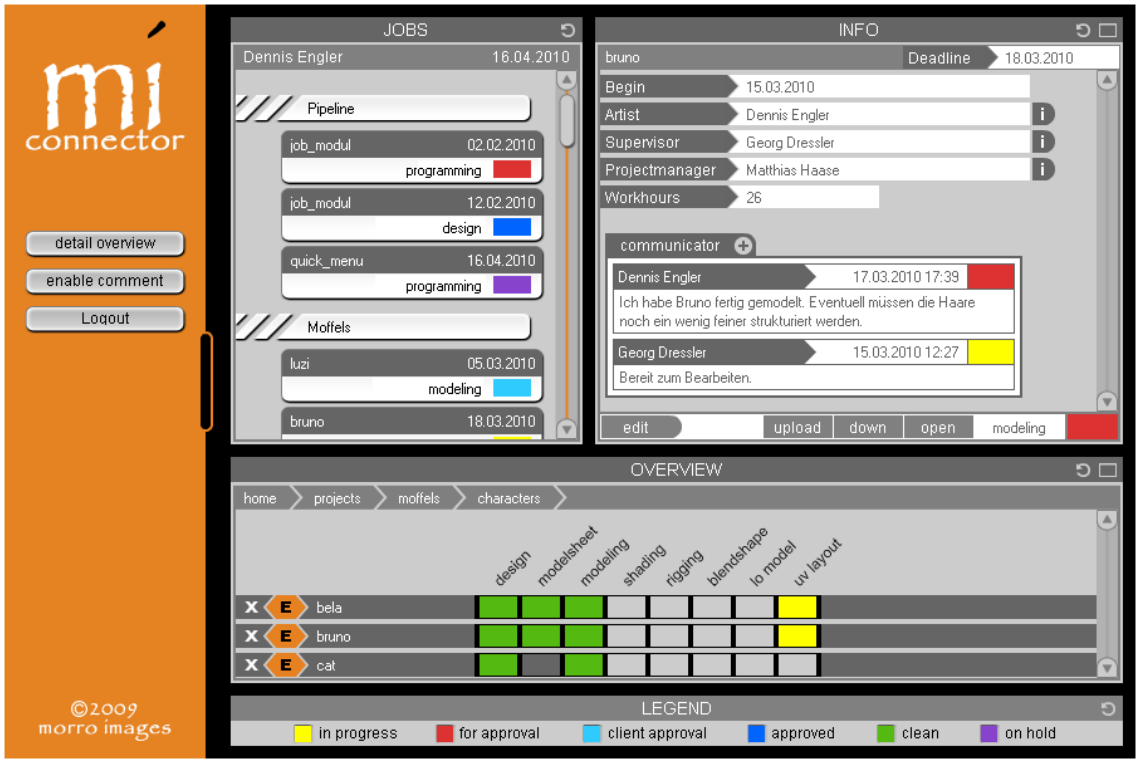


Abb. 40) inhaltlicher und grafischer Aufbau der Applikation[64]

Der Aufbau der GUI gestaltet sich sehr ähnlich wie im Kapitel *Pflichtenheft* erläutert (siehe Abb. 40). Auf der linken Seite befindet sich die Navigationsleiste, oben mittig das JobView-Modul und oben rechts das InfoView-Modul. Unterhalb der beiden Module

[64] eigene Darstellung, komplette Darstellung der Applikation

befindet sich das OverView-Modul, das die tabellarische Ansicht für die Projektstrukturen bereit hält. Im Folgenden findet eine genaue Erläuterung zum inhaltlichen und grafischen Aufbau der bestehenden Module statt.

Das JobView-Modul:

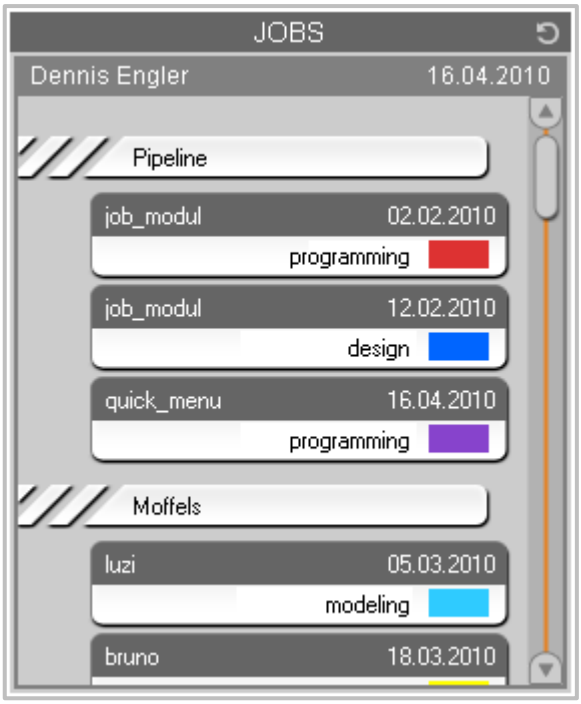


Abb. 41) inhaltlicher und grafischer Aufbau des JobView-Moduls[65]

Das Hauptaugenmerk bei der Gestaltung und Programmierung des JobView-Moduls lag darauf, dem Benutzer nach der Anmeldung einen umfassenden Überblick über seine aktuell zu erledigenden Aufgaben zu gewähren. Die angezeigten Jobs sind nach Projekten gruppiert, wodurch eine rasche Zuordnung gewährleistet ist. Verschiedene Informationen wie der aktuelle Status und die Deadline sind im Objekt platziert. Sie sind darüber hinaus auch als Schaltflächen konzipiert. Mittels Mausklick werden alle Informationen ohne Umwege im InfoView-Modul angezeigt und können bei Bedarf sofort bearbeitet werden. Sobald eine Aufgabe durch den Projektleiter geschlossen wurde, verschwindet der entsprechende Eintrag aus dem JobView-Modul und gilt als erledigt. Über eine kleine Schaltfläche auf der rechten Seite, oberhalb des Statusbalkens lässt sich die Liste manuell aktualisieren.

[65] eigene Darstellung, Ausschnitt aus der Applikation

Das InfoView-Modul:

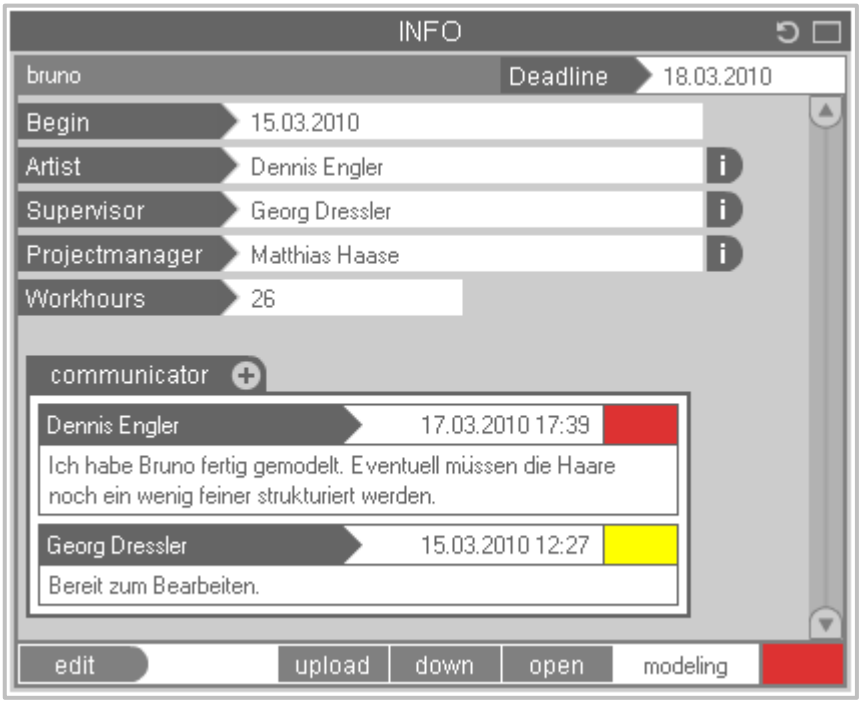


Abb. 42) inhaltlicher und grafischer Aufbau des InfoView-Moduls[66]

Das InfoView-Modul dient der Anzeige aller Informationen eines Jobs. Dazu gehören sowohl das Startdatum als auch die Deadline der entsprechenden Aufgabe. Weiterhin sind alle Mitarbeiter aufgelistet, die an der Bearbeitung der Aufgabe beteiligt sind. Hierzu gehören der Artist, der den Job zu erledigen hat, genauso wie sein Supervisor der für die Abnahmen zuständig ist. Der Projektleiter überwacht schließlich alle Prozesse. Über kleine Schaltflächen hinter den Einträgen können, abhängig von den Rechten des Users, alle Informationen zu dem jeweiligen Mitarbeiter abgerufen werden. In diesem Fall werden alle Daten bezüglich des Jobs ausgeblendet. Auch die Schaltfläche *edit* verschwindet, an ihrer Stelle taucht ein Button auf, mittels dessen der Benutzer zurück zum zuvor angezeigten Job kommt. Innerhalb des InfoView-Moduls ist auch der *Communicator* untergebracht, dessen Funktion bereits erläutert wurde (vgl. Kapitel 4.1.4). An dieser Stelle sei noch angemerkt, dass der jeweils oberste Eintrag zugleich auch der aktuellste ist. Über die drei unten, mittig platzierten Schaltflächen *upload*, *down* und *open* erfolgt die Steuerung der Mediendateien. Rechts daneben finden sich der Name des Jobs und sein aktueller Status.

[66] eigene Darstellung, Ausschnitt aus der Applikation

Das Overview-Modul:

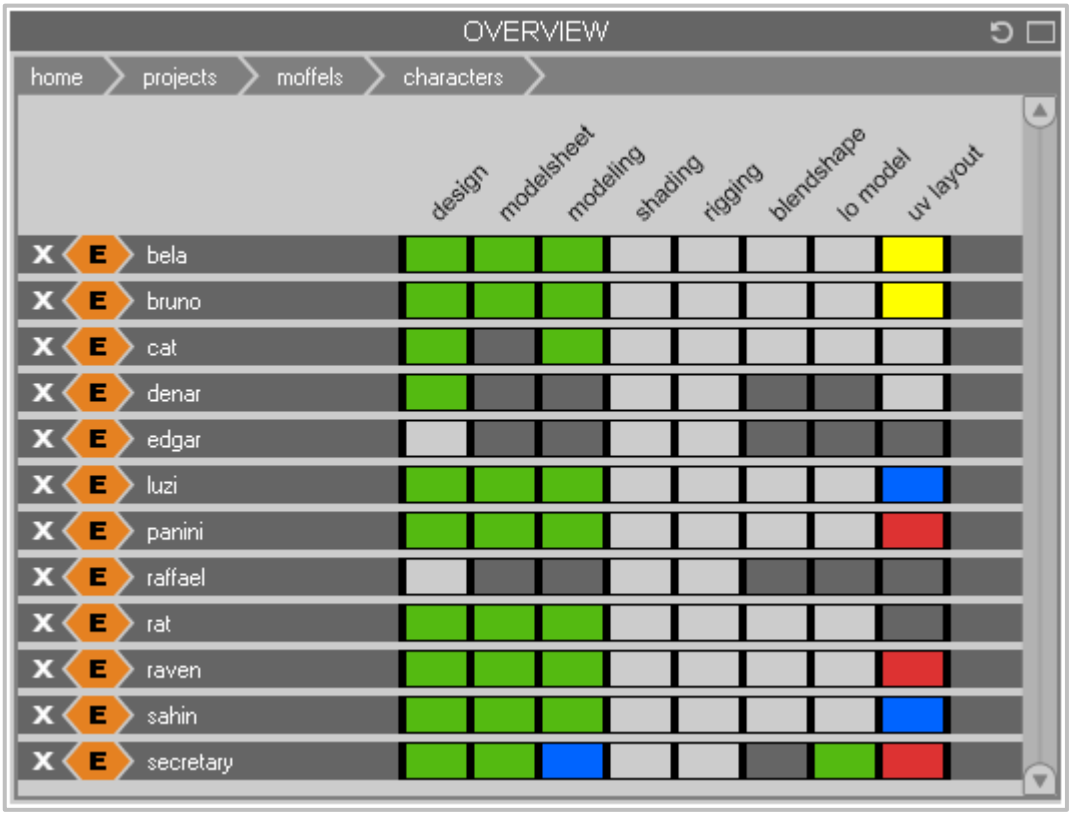


Abb. 43) inhaltlicher und grafischer Aufbau des Overview-Moduls mit detaillierter Ansicht[67]

Dieses Modul wahrt die geforderte Übersicht über alle laufenden Prozesse. Es soll hauptsächlich Verwendung in der Navigation durch die Projekte finden. Projektleiter und Supervisor können sich übersichtlich einen Großteil der von ihnen benötigten Informationen anzeigen lassen. Über die beiden Schaltflächen am linken Rand jeder Zeile kann der jeweilige Eintrag gelöscht oder editiert werden. Diese Funktion steht nur jenen Benutzern zur Verfügung, welche über die entsprechenden Rechte verfügen. Fehlen diese Rechte, so werden die Schaltflächen ausgeblendet. Am oberen Rand befindet sich eine Navigationsleiste die zugleich dazu dient, die aktuelle Position innerhalb der Projektstruktur wiederzugeben. Mittels Mausklick gelangt der Benutzer in die ausgewählte Hierarchieebene.

[67] eigene Darstellung, Ausschnitt aus der Applikation

Das Quick-Menü:

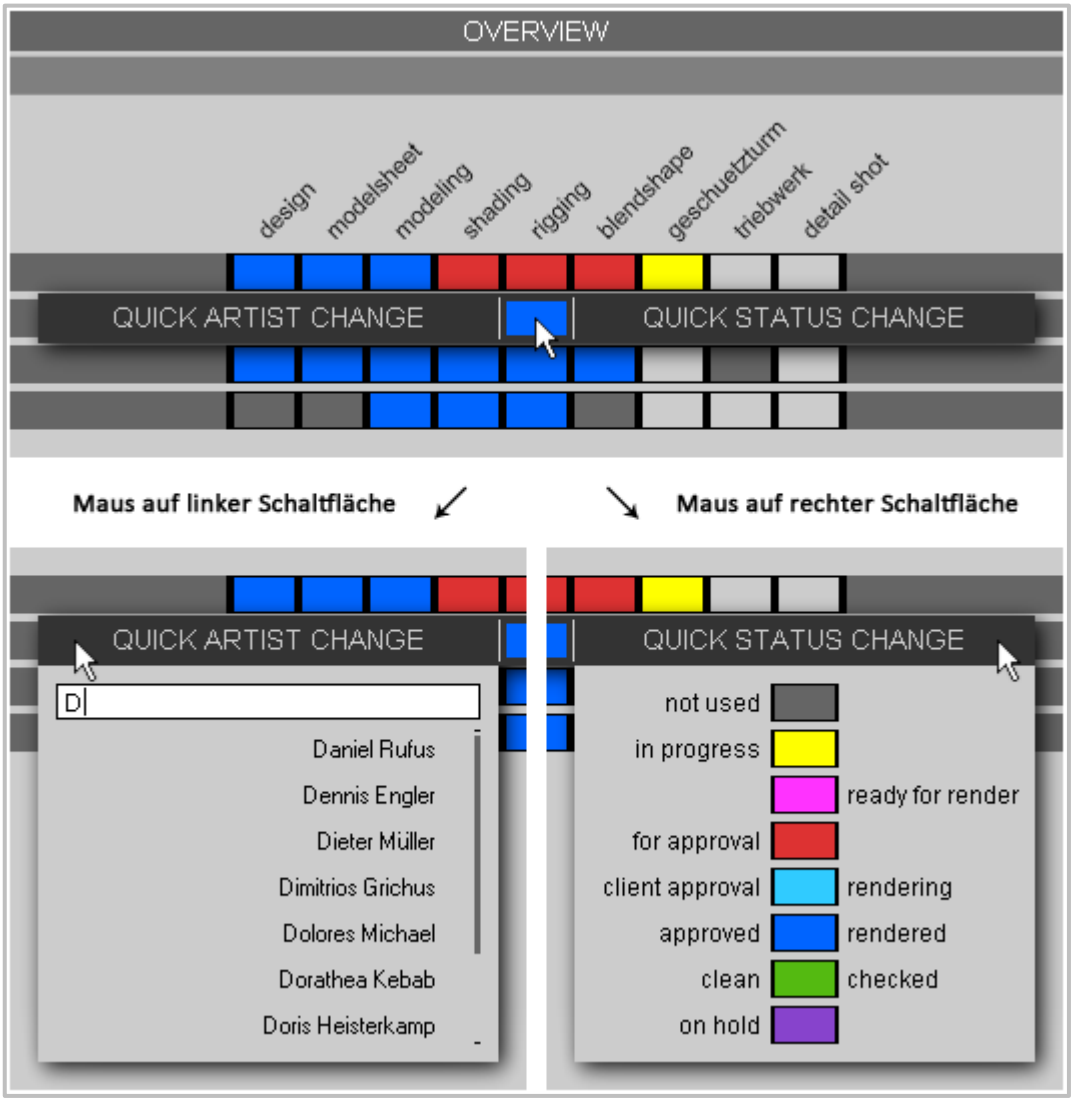


Abb. 44) Funktionsweise des Quick-Menüs[68]

Diese Funktion wurde ursprünglich für die Projektleiter geschaffen, um den Status von Aufgaben schneller zu ändern oder ihnen Mitarbeiter zuzuweisen. Im Nachhinein wurde das Quick-Menü leicht verändert um auch eine zügige Navigation innerhalb des OverView-Moduls zu gewährleisten. Die Funktionsweise gestaltet wie folgt: Sobald ein Benutzer mit der Maustaste auf ein Status-Feld klickt wird die *quickMenu* Klasse geladen. Entscheidend ist nun wo er die Maustaste loslässt. Erfolgt der in Programmiersprachen üblicherweise als *Maus-Release* bezeichnete Befehl auf dem Statusfeld, so wird die entsprechende Aufgabe im Info-Modul geladen. Wenn bei gedrückter Maustaste mit dem Mauszeiger auf die rechte Schaltfläche *Quick Status*

[68] eigene Darstellung, Bildmontage aus verschiedenen Ausschnitten aus der Applikation

Change navigiert und die Maustaste hierüber losgelassen wird, öffnet sich ein Menü. Die eingeblendeten Statusfelder sind als Schaltflächen konzipiert. Über einen erneuten Mausklick auf eines der Elemente wird der Status sofort geändert. Ähnlich funktioniert das System wenn man den Mitarbeiter ändern möchte. Hierfür muss die Maustaste allerdings über der linken Schaltfläche *Quick Artist Change* losgelassen werden. Daraufhin öffnet sich ein Menü mit einem Eingabefeld und einer darunter platzierten Liste der Mitarbeiter. Um nicht jedes Mal die gesamte Liste nach dem gewünschten Eintrag durchsuchen zu müssen, wurde eine automatisch vervollständigende Suche integriert. Sobald ein Buchstabe in das Eingabefeld eingetippt wird, sorgt eine Filterfunktion dafür, dass lediglich die dazu passenden Elemente ausgegeben werden.

5 Schlussbetrachtung

5.1 Zusammenfassung

Das Ziel der Arbeit war, durch einen implementierten Anwendungsprototyp die Arbeitsprozesse der morro images GmbH & Co. KG, in Form von Kommunikations- und Informationsaufwendungen, zu beschleunigen. Darüber hinaus wurde das Ziel gesetzt, eine sichere Anbindung über das Internet zu gewährleisten, um den Projektleitern die Möglichkeit einzuräumen, ihre Aufträge auch unterwegs überwachen zu können.

Um diese Ziele zu erreichen, wurde zunächst eine sorgfältige Erörterung der heutzutage typischen Anforderungen an eine solche branchenspezifische Managementsoftware durchgeführt (vgl. 2.1). Es konnten einerseits das *Prinzip der Organisation* und andererseits das *Prinzip der Verwaltung* als zwei grundlegende Möglichkeiten des Managements in Medienproduktionsunternehmen ermittelt werden (vgl. 2.2). Anhand einer genaueren Untersuchung dieser beiden Ansätze ließen sich hinreichend viele Informationen für eine Konzeptionsphase sammeln. Dazu wurden am Beispiel von AceProject (vgl. 2.3) und Alienbrain (vgl. 2.4) – zwei gewichtigen Vertretern dieser Prinzipien – die Vor- und Nachteile solcher Systeme aufgezeigt. In die Untersuchung flossen hierzu auch benutzerspezifische Kriterien ein, nach denen später das System zur Rechteverwaltung konzipiert wurde.

Auf Basis der gewonnenen Erkenntnisse wurden im Kapitel 3.1 konkrete programmiertechnische Ziele definiert. Diese wurden anhand der bis dato im Unternehmen eingesetzten Pipeline erarbeitet. Im nachfolgenden Kapitel fand eine Erläuterung der zum Erreichen der Ziele erforderlichen Technologien statt (vgl. 3.2). Auf diesen Technologien beruhend wurde im Kapitel 3.3 eine Herangehensweise für die programmiertechnische Umsetzung dargelegt und erläutert. Dieses Pflichtenheft diente gleichzeitig als Grundlage für die Implementierung der eigenen Anwendung

In Kapitel 4 wurde schließlich die Umsetzung der programmiertechnischen Arbeitsschritte beschrieben. Dazu wurde zunächst das verwendete DBMS näher beleuchtet (vgl. 4.1.1), bevor die Definition einer geeigneten Datenbankstruktur stattfinden

konnte (vgl. 4.1.3). In Abschnitt 4.1.4 wurde diese Struktur anhand der Funktionsweise der Tabellen verdeutlicht. In Kapitel 4.2 wurde einerseits die Verwendungsweise der FileMaker API für PHP untersucht und andererseits ihre Implementierung anhand eines Beispiels beschrieben. Da die Anbindung an eine Datenbank ein zentraler Kern dieser Arbeit war, kam diesen beiden Kapiteln (vgl. 4.2.1 und 4.2.2) eine große Gewichtung zu. Schließlich erfolgte eine Beschreibung der Vorgehensweise zur Implementierung des Frontends mit Hilfe von ActionScript. Dazu wurden in Kapitel 4.3.1 einige Grundlagen zum Umgang dieser objektorientierten Programmiersprache abgehandelt. Aufgrund des immensen Umfangs der Applikation wurde anhand einiger ausgewählter Beispiele die Vorgehensweise der Implementierung dargelegt (vgl. 4.3.2). Es folgte die Erklärung eines selbst konzipierten Systems zur Rechteverwaltung (vgl. 4.4). Im abschließenden Kapitel 4.5 wurde zunächst die Verwendung einer geeigneten Farbpalette diskutiert, bevor auf die gestaltungstechnischen Aspekte, hinsichtlich des Layouts, eingegangen wurde.

5.2 Fazit und Ausblick

Der entwickelte Anwendungsprototyp konnte erfolgreich in die bestehende Systemstruktur integriert werden. Mittlerweile laufen bereits alle Medienproduktionen innerhalb des Unternehmens über die neue Postproduction-Pipeline. Da die Mitarbeiterzahl während des Entstehungsprozesses dieser Arbeit rasant auf das Doppelte angestiegen ist, kann auch die Verträglichkeit mit hohem Datenaufkommen bestätigt werden. Bislang kam es zu keinerlei Ausfällen und die Geschwindigkeitsreserven des Systems sind mit derzeit etwa 25 Benutzern bei weitem nicht ausgereizt. Die zuvor verwendete Pipeline des Unternehmens wäre hier aufgrund eines größeren Projekts vielleicht an ihre Grenzen geraten. Da die Umstellung jedoch schon vorher stattfand, lässt sich dies nicht endgültig klären. Externen Mitarbeitern ist es nun möglich ihre Daten auch von außerhalb in das System einzubinden. Damit einher geht eine enorme Zeitersparnis, weil diese Mitarbeiter nun, beispielsweise bei Abnahmen, nicht mehr zwingend persönlich im Unternehmen sein müssen.

Für die positive Annahme der Applikation innerhalb der Projektteams, zeichnet der Autor insbesondere die Analyse marktführender vergleichbarer Systeme verantwortlich. So konnten aufschlussreiche Ergebnisse aus der Untersuchung gewonnen werden, welche direkt in die Entwicklung der eigenen Anwendung fließen konnten. Die ausgeprägte Konzeptionsphase half dabei während der Programmierung richtige Prioritäten setzen zu können. Hier wäre im speziellen noch einmal die im Voraus erstellte Previsualisierung des Layouts zu nennen, welche bei der Gestaltung der Anwendung einen zügigen Workflow mit sich brachte.

Im Verlauf der Bearbeitung des schriftlichen Teils wurde bereits eine Vielzahl der vorhandenen Fehler im Programm behoben. Darüber hinaus ist die Konzeptionsphase der *Version 2.0* genannten Anwendung bereits in Planung. Das Hauptaugenmerk wird hierbei auf einer direkten Verknüpfung mit gängigen Produktionsanwendungen liegen. Sie soll den Mitarbeitern noch mehr Arbeit abnehmen, indem Dateien innerhalb der Postproduction-Pipeline geöffnet werden und beim Speichern eine automatische Versionierung vorgenommen wird.

Desweiteren soll die Modularität noch weiter in den Vordergrund rücken. Geplant ist einerseits, dass die bereits vorhanden Module noch unabhängiger voneinander werden und die Anwendung andererseits noch interaktiver wird. So sollen sich die Module beliebig hinzu schalten bzw. ausblenden lassen. Darüber hinaus sind weitere Module, wie beispielsweise ein Kalender, vorgesehen. Er soll es ermöglichen, dass Personalmanager einen genauen Überblick erhalten, welcher Mitarbeiter über welche Zeiträume an welchen Projekten arbeitet. Einen Ausblick auf einige der angedachten Funktionen liefert eine vorläufige Previsualisierung (siehe Abb. 45 auf Seite 73, aus Gründen der besseren Übersicht wird diese im Querformat dargestellt).

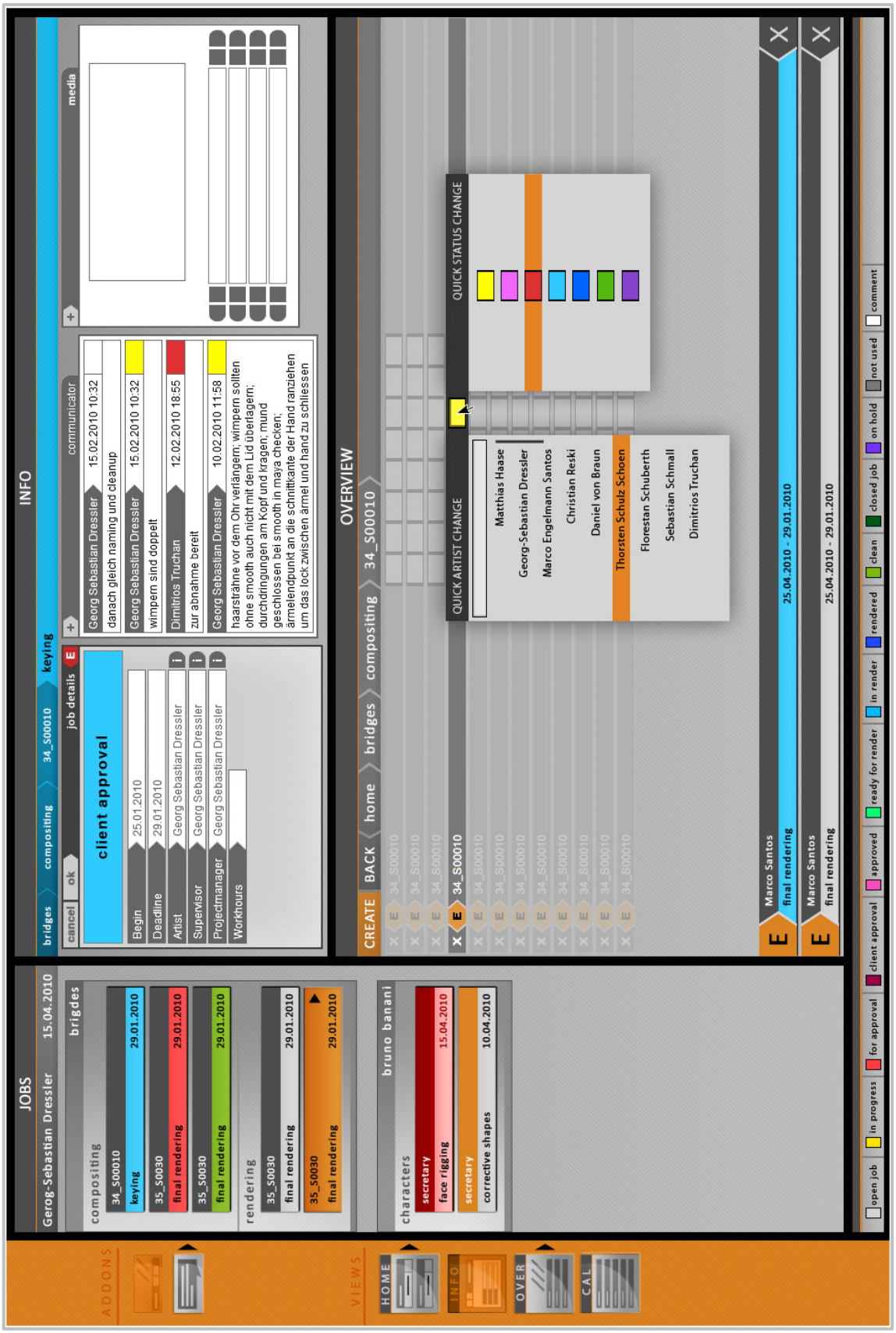


Abb. 45) Previsualisierung der Postproduction-Pipeline Version 2.0[69]

[69] eigene Darstellung

Abbildungsverzeichnis

Literaturverzeichnis

Balzert, Heide: "Basiswissen Web-Programmierung: XHTML, CSS, JavaScript, XML, PHP, JSP, ASP.NET, Ajax", 1. Auflage, W3L GmbH, Herdecke 2007

Braunstein, Roger ; Wright, Mims H. ; Noble, Joshua J.: "ActionScript™ 3.0 Bible", Wiley Publishing Inc., Indianapolis 2008

Kleinschmidt, Peter ; Rank, Christian: "Relationale Datenbanksysteme: Eine praktische Einführung", 3. Auflage, Springer-Verlag Berlin, Heidelberg 2005

Meier, Andreas: "Relationale Datenbanken: Leitfaden für die Praxis", 5. Auflage, Springer-Verlag Berlin, Heidelberg 2004

Moos, Alfred: "Datenbank-Engineering: Analyse, Entwurf und Implementierung objektrelationaler Datenbanken", 3. Auflage, Vieweg-Verlag, Wiesbaden 2004

Natschew, Claudia: " Media Asset Management: Ein Überblick", In: Digital Production, München: ACT GmbH, 03/2004, Seite 107 - 110

Internetquellen

Blassl, Horst ; Früchtel, Harald (Vortrag): "Das neue FileMaker Forum: Einsatz von Web 2.0, FM PHP API und XML", http://downloads.filemaker.de/konferenz2006/FileMakerForum_Fruechtel_Blassl.pdf, 20.05.2010

FileMaker (Handbuch): "FileMaker Server 11: Custom Web Publishing mit PHP", http://filemaker.de/support/product/docs/fms/fms11_cwp_php_de.pdf, 19.05.2010

Stark, Jonathan ; Hansen, Chris: "Web Publishing with FileMaker and PHP", http://jonathanstark.com/web-publishing-with-filemaker-and-php.php, 20.05.2010

Selbstständigkeitserklärung

Hiermit erkläre ich, Dennis Engler, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Mittweida, den 31. Mai 2010